UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

Cuadros de mando sobre Moodle

Autor/es

LAURA MALLAGARAY CORRAL

Director/es

CÉSAR DOMÍNGUEZ PÉREZ

Facultad

Facultad de Ciencia y Tecnología

Titulación

Grado en Ingeniería Informática

Departamento

MATEMÁTICAS Y COMPUTACIÓN

Curso académico

2016-17

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

Facultad de Ciencia y Tecnología

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Cuadros de mando sobre Moodle

Alumno:

Laura Mallagaray Corral

Tutores:

César Domínguez Pérez

Logroño, Febrero, 2017

Agradecer a mis padres y familiares el apoyo moral y económico
que me han dado a lo largo de toda la carrera.
A todos mis profesores del Grado que me han aportado tanta
formación, y en especial, a mi tutor del Trabajo Fin de Grado,
César Domínguez Pérez, por su esfuerzo y dedicación.

Resumen

El objetivo de este trabajo es posibilitar un estudio interactivo y multidimensional, de forma sencilla y visual, de determinados datos almacenados en Moodle. De esta forma, los centros podrán sacar provecho de ellos y utilizar esta información para mejorar su funcionamiento y el proceso de enseñanza-aprendizaje de sus alumnos. Para ello, realizaremos un proceso de extracción, transformación y carga de datos en un Data WareHouse, que acabará con la creación de un cubo multidimensional. El usuario podrá realizar consultas a este mediante la herramienta Pentaho. Posteriormente, crearemos una aplicación web mediante la cual se podrá acceder a diferentes cuadros de mando que realizaremos, así como descargarlos. Todos ellos serán referentes a la información almacenada en el cubo.

Abstract

The aim of this work is enabling an interactive and multidimensional study of certain data stored in Moodle. This will be able to be done in a simple and visual way. Therefore schools will be able to use this information to improve its functioning and the teaching-learning process of its students. So we will carry out a process of extraction, transformation and data loading in a Data Warehouse, which will end with the creation of a multidimensional cube. Users will be able to consult it through Pentaho. Later we will create a web application which will let users access different dashboards that we will have carried out as well as download them. All of them will be referring to the information stored in the cube.

Índice general

Introducción

Business Intelligence (BI) es la habilidad para transformar los datos en información, y la información en conocimiento, de manera que así se pueda observar y comprender lo que está ocurriendo, predecir lo que ocurrirá y a partir de ello, tomar las decisiones más óptimas en cada ocasión. Esto es realmente útil principalmente en el mundo de los negocios y de hecho, su propio nombre hace referencia a ello («Inteligencia empresarial»).

Por tanto, se conocen como BI a aquellas metodologías, aplicaciones y tecnologías que posibilitan esta transformación, es decir, como bien se dice en [9], que permiten reunir, depurar y transformar datos de diferentes sistemas transaccionales e información desestructurada (bases de datos, ficheros de texto, etc), en información estructurada, para su explotación directa o para su análisis y conversión en conocimiento. Y es que, como bien decía ya Edgar Frank Codd a principios de los noventa, disponer de un sistema de bases de datos relacionales no significa contar con un soporte directo para la toma de decisiones, ya que muchas de estas se basan en un análisis de naturaleza multidimensional, es decir, en analizar las diferentes relaciones existentes entre diversos indicadores.

En la figura 1 podemos observar la serie de procesos cuya ejecución nos lleva de los datos al conocimiento. Notar que se denomina Data Warehouse, tal y como se dice en [5], a una base de datos corporativa caracterizada por integrar y depurar información de una o más fuentes distintas, para luego procesarla y explotarla mediante una herramienta OLAP permitiendo así su análisis desde infinidad de pespectivas y con grandes velocidades de respuesta. En particular, dicho concepto surgió a raíz de las dificultades de los sistemas tradicionales en satisfacer las necesidades informacionales (ver diferencias entre los sistemas tradicionales y los Data Warehouses en el anexo 1). Además, en la mayoría de las ocasiones, su creación representa el primer paso, desde el punto de vista técnico, para implantar una solución completa y fiable de Business Intelligence. Por otra parte, a la fase de extracción, elaboración o transformación y carga de los nuevos datos en el Data Warehouse, se le conoce como proceso ETL («Extract, Transform and Load»).

Los principales productos de Business Intelligence que existen actualmente son: Cuadros de

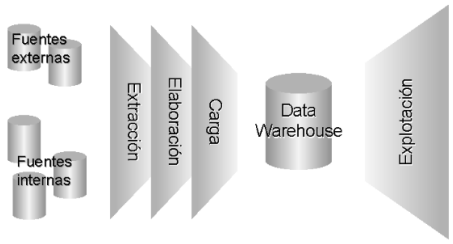


Figura 1: Procesos que conforman la fase de transformación de los datos en conocimiento.

Mando Integrales (CMI), Sistemas de Soporte a la Decisión (DSS) y Sistemas de Información Ejecutiva (EIS).

En este proyecto nos centraremos en los CMI, también conocidos como Balanced Scorecards (BSC) o dashboards, los cuales tal y como se dice en [6], se tratan de herramientas de gestión que permiten monitorizar mediante indicadores, el cumplimiento de la estrategia desarrollada por la dirección, a la vez que tomar decisiones rápidas y acertadas para alcanzar sus objetivos. En particular, implementaremos diferentes cuadros de mando integrales partiendo de datos almacenados en Moodle (plataforma para la enseñanza y organización escolar a través de la cual los estudiantes resuelven exámenes, envían tareas al profesor, consultan sus notas...) y tras llevar a cabo sobre ellos el proceso ETL correspondiente, la construcción del Data WareHouse indicado, y la explotación del mismo obteniendo así un modelo multidimensional (en concreto, un cubo) de la información contenida en él, que posibilita el análisis interactivo de la misma.

En dichos cuadros de mando, podremos observar ciertas relaciones existentes entre diversos factores: calificaciones en función de las asignaturas, de los años o de los centros en los que se imparten, número de alumnos matriculados por asignatura cada año en un centro... De esta forma, los centros dispondrán de un análisis exhaustivo del recorrido académico de cada uno de sus alumnos, el cual será interesante compartir con sus respectivos familiares, y de todos ellos en conjunto, pudiendo identificar así aspectos influyentes tanto positivamente como negativamente en su rendimiento, comparar la eficacia de diferentes métodos de evaluación o de enseñanza, realizar comparaciones con otros centros, etc. Por lo que, teniendo en cuenta dichos estudios, les será más fácil tomar las decisiones más convenientes de cara a conseguir que el proceso de enseñanza-aprendizaje de cada uno de sus alumnos sea el mejor posible.

En esta memoria, comenzaremos plasmando la planificación del proyecto realizada inicialmente para asegurarnos del éxito del mismo y que incluye los siguientes planes: del alcance (objetivos, estudio de alternativas, entregables y EDT), de las tareas (orden de realización, diagrama de Gantt, de hitos y tiempo estimado de cada una) y de gestión del proyecto (gestión de comunicación, de riesgos, de calidad y de cambios).

Posteriormente, explicaremos las fases de análisis de requisitos y de diseño realizadas. Estas aparecerán subdivididas en dos partes ya que han sido realizadas en dos ocasiones distintas pues así lo requería el cliente: la primera justo al acabar la planificación, y la segunda una vez habíamos construido el cubo pertinente, tras reunirnos otra vez con el cliente.

A continuación, explicaremos aspectos sobre el proceso de implementación seguido, es decir, del desarrollo del proyecto: pasos que hemos ido realizando, problemas que nos han ido apareciendo y cómo los hemos conseguido resolver, etc.

Y finalmente, incluiremos el informe de seguimiento que hemos ido realizando a lo largo del transcurso del proyecto (control de calidad y de cambios) y las lecciones que hemos aprendido a lo largo de ese tiempo.

Capítulo 1

Planificación del proyecto

1.1. Alcance

1.1.1. Objetivos

Tras la primera reunión con el cliente, quedaron establecidos dos objetivos claros en el proyecto.

Por un lado, abstraer los datos que se almacenan en Moodle, cuyo análisis interactivo resulta de interés (medidas o factores que son interesantes estudiar y en función de qué dimensiones o aspectos) del lugar en el que están almacenados, ya que los usuarios deben saber qué buscar pero no es su responsabilidad saber dónde. En particular, construir un cubo multidimensional partiendo del Data WareHouse correspondiente creado previamente y cuya información sea actualizada automáticamente cada mediodía, que el usuario pueda consultar para obtener así, de forma rápida y fiable, los datos almacenados en Moodle correspondientes a la medida y dimensiones que elija. Por ejemplo, si el usuario elige como medida la nota media y como dimensiones la asignatura y el alumno, obtendrá las notas medias de cada uno de los alumnos en cada una de las asignaturas en las que están matriculados. Además, también podrá filtrar estos resultados según determinados factores, que deberán ser dimensiones en el cubo. Volviendo al ejemplo anterior, si por ejemplo elegimos que alumno sea un filtro, solo se mostrarán las notas medias del alumno que indiquemos, en las asignaturas en las que está matriculado.

Estas consultas se realizarán mediante una herramienta BI (después estudiaremos distintas alternativas para elegir la más apropiada). De forma que así, evitaremos que el usuario tenga que tener conocimientos de MySQL, que es el gestor de base de datos que utilizaremos en todo el proyecto, y que tenga que conocer dónde se encuentran los distintos datos.

Y por otro lado, el segundo objetivo de este proyecto es la visualización gráfica del resultado de algunas de las consultas comentadas anteriormente, a través de una aplicación web. Es decir, de mostrar visualmente las medidas en base a una o más de las dimensiones disponibles, pudiendo así identificar las relaciones existentes entre ellas. Se busca que así, los centros saquen más provecho de los datos almacenados en Moodle, pudiendo mostrar a los familiares de sus alumnos, o incluso a los propios alumnos, aspectos referentes a la evolución de su rendimiento académico, entender por qué está ocurriendo algo, predecir qué ocurrirá en el futuro, y tomar decisiones fundamentadas en estos análisis de cara a conseguir que tanto el proceso de enseñanza-aprendizaje de cada uno de sus alumnos como el funcionamiento del centro en general, sea el mejor posible. Técnicamente hablando, este objetivo se basa en la implementación de los cuadros de mandos integrales respectivos a las relaciones que se desean estudiar y de una aplicación mediante la cual se pueda acceder a ellos. Se realizarán tantos CMI como las limitaciones de

tiempo lo permitan y en el orden correspondiente a las prioridades del cliente.

En resumen, el producto final se tratará de una máquina virtual Ubuntu 16.04, que tenga instalados Moodle, MySQL y la herramienta BI elegida, en la cual se encuentre todo lo necesario para poder realizar fácilmente, mediante la herramienta BI, las consultas citadas anteriormente, y para poder visualizar mediante la aplicación web, todos los CMI que hemos implementado. Además, se entregará al cliente un manual de configuración para guiarle en caso de que quiera disponer de esta funcionalidad en otro lugar, etc.

En el apartado 2 explicaremos con mayor profundidad los requisitos que desea el cliente: medidas, dimensiones y filtros que le interesan, su prioridad en cuadros de mando a realizar, interfaz que desea, etc.

1.1.2. Estudio de alternativas

A pesar de haber estado trabajando ya con Pentaho en las prácticas y por tanto saber que esta es muy apropiada para la realización de este proyecto, vamos a realizar un estudio de la misma y de otras de las plataformas de Bussiness Intelligence, con la finalidad de asegurarnos que elegimos la más apropiada. La única condición que nos impone el cliente es que esta sea gratuita ya que, en principio, no está dispuesto a destinar nada de su presupuesto a este fin.

En la figura 1.1, presente en [10], podemos observar una comparación de las características fundamentales de las mismas a la hora de decantarnos por una de ellas. En ella se valoran los siguientes aspectos: permite procesos ETL, incluye soluciones OLAP, incluye generación de informes (G.I.), permite crear Dashboards (DBd), permite crear Scorecards (SCd), dispone de servicio técnico (IT), tiene componentes adicionales (Comp.), depende de otras herramientas (Dep.), licencia Open Source (OpS., M = Mixta), dificultad de encontrar información (D.Info, A=alta, M=media y B=baja), dispone de interfaz gráfica (GUI), permite exportar datos (Export.) y valoración global de 0 a 5 (verde = +1, amarillo = + 0.5).

Nombre	ETL	OLAP	G.I.	DBd	SCd	IT	Comp.	Dep.	OpS	D.Info.	GUI	Export.	Valoración
OlapCube	No	Si	No	No	No	Si	Si	No	No	A	Si	Si	
OlapCube Dashboard	No	Si	No	Si	No	Si	Si	Si	Si	A	Si	Si	
Cube-it.Zero	No	Si	No	No	No	Si	Si	No	M	M	Si	Si	
Mondrian	No	Si	No	No	No	No	Si	No	Si	B	No	Si	
OpenI	No	Si	Si	Si	No	No	No	Si	Si	M	Si	Si	
jOLAP	No	Si	No	Si	No	Si	Si	No	Si	B	Si	Si	
Jedox	Si	Si	Si	Si	No	Si	Si	No	No	B	Si	Si	
JPivot	No	Si	No	No	Si	No	No	Si	Si	A	Si	No	
InstantOLAP	No	Si	Si	Si	Si	No	No	Si	Si	M	Si	Si	
OLAP Browser	No	Si	No	No	No	Si	No	Si	No	M	Si	Si	
phpMyOLAP	Si	No	Si	No	No	No	No	Si	Si	A	Si	Si	
RapidAnalytics	No	No	Si	Si	No	No	No	Si	Si	B	Si	Si	
Qlik Sense	No	No	Si	Si	Si	Si	Si	No	M	M	Si	Si	
QlikView	No	No	Si	Si	Si	Si	Si	No	Si	B	Si	Si	
InetSoft	No	No	Si	Si	Si	Si	No	No	Si	B	Si	Si	
Marvelit	No	No	Si	Si	Si	Si	No	No	No	A	Si	Si	
Pentaho	Si	Si	Si	Si	Si	Si	Si	No	M	B	Si	Si	

Figura 1.1: Comparación herramientas BI.

Finalmente, nos decantamos rotundamente por Pentaho, ya que además de estar familiarizados con esta herramienta, es la que ha conseguido mayor puntuación en el estudio anterior, disponiendo incluso de más características de las que necesitamos, lo cual nos podrá venir bien en el futuro para ampliar o modificar este proyecto.

1.1.3. EDT, descripción de tareas y entregables

Hemos desarrollado esta EDT (ver figura 1.2) teniendo en cuenta que las tareas que deberemos realizar para este proyecto son las siguientes:

- **P1.1 Planificación**: Esta tarea consistirá en crear una planificación que se seguirá a lo largo del proyecto y que marcará los plazos, objetivos a conseguir y tareas a realizar para su consecución. También incluirá planes de comunicación, de riesgos, de calidad y de gestión de cambios. Por tanto, el resultado de esta será un entregable interno al proyecto, que consistirá en la planificación redactada, y que será incluido en la memoria.
- **P1.2 Análisis de requisitos**: Recogida de requisitos del proyecto que tendrá como resultado otro entregable interno a incluir en la memoria.
- **P1.3 Seguimiento**: Engloba todas las actividades de seguimiento y control del proyecto. Se anotará el tiempo empleado para cada tarea y su periodo real de realización, se compararán con las estimaciones realizadas analizando el por qué de las desviaciones, y se realizará un seguimiento de la comunicación, la calidad, los riesgos, y los cambios. Todo ello se redactará y conformará otro entregable interno, que formará parte de la memoria final.
- **P1.4 Lecciones aprendidas**: Redacción de las lecciones aprendidas a lo largo del desarrollo del proyecto de cara a no cometer los mismos errores en el futuro. Será otro entregable interno y que añadiremos después a la memoria.
- **P1.5 Reuniones**: A lo largo del proyecto se llevarán a cabo varias reuniones: con el cliente, para acordar y aceptar los objetivos del proyecto y sus requisitos, así como para mostrarle finalmente el producto (en principio cuatro de media hora), y con el tutor del proyecto para irle mostrando lo trabajado hasta el momento, recibir sugerencias o consultarle dudas (se estima que una de una hora cada dos semanas). Esta tarea también incluirá la redacción de las actas de las reuniones establecidas, las cuales formarán otro entregable interno que será incluido en los anexos de la memoria.
- **P2.1.1 Estudio Moodle**: Una vez tenemos instalado Moodle, investigación sobre su funcionamiento, datos posibles a almacenar, lugares donde se almacenan: estructura de su base de datos, tablas en las que aparecen los diferentes datos, etc. Y finalmente, redacción de las medidas y dimensiones que pueden resultar interesantes tener de cara a realizar cuadros de mando integrales, así como de algunos de los CMI que se pueden realizar con ellas. El resultado de esta tarea será un entregable interno que será presentado al cliente en una reunión para que este elija lo que desea y fije así los requisitos del proyecto, y un resumen de la estructura de la base de datos de Moodle.
- **P2.1.2 Introducción datos**: Debido a que no disponemos de ninguna base de datos poblada de Moodle, deberemos preparar el entorno de Moodle a partir del cual trabajaremos, ya que necesitamos datos de prueba para ir realizando y probando nuestro proyecto. Por lo que en lo que respecta a esta tarea, introduciremos cursos, usuarios, exámenes, calificaciones, tareas, etc. Por lo que esta tarea tendrá como resultado la base de datos de Moodle poblada de datos a partir de la cual podremos trabajar, que será un entregable interno al proyecto.

- **P2.2 Formación CMI**: Se prevee que se realizarán diferentes CMI usando el CDE («*Community Dashboard Editor*») de Pentaho o programándolos con ayuda de Highcharts y habiendo establecido previamente una conexión con el cubo. Por lo que esta tarea incluye el aprendizaje de todo ello junto con la realización de algunos ejemplos, los cuales serán entregables internos cuya consulta posteriormente nos será útil en la implementación.

- **P3.1 Diseño**: Diseño del Data WareHouse, del proceso ETL, del cubo, de las consultas, de los CMI, de la aplicación web a través de la cual el usuario podrá visualizar los cuadros y de las pruebas a realizar. Todo ello se irá redactando en un entregable interno que será incluido posteriormente en la memoria.

- **P3.2.1 Construcción Data WareHouse**: Creación del Data WareHouse y de su estructura (tablas de dimensiones y de hechos, con sus correspondientes campos) teniendo en cuenta las medidas y dimensiones decididas anteriormente. El resultado de esta tarea será el Data WareHouse, y por tanto, un entregable interno.

- **P3.2.2 Proceso ETL**: Esta tarea consiste en llevar a cabo el proceso ETL partiendo de los datos almacenados en la base de datos de Moodle que tenemos y, usando Kettle, realizar las transformaciones necesarias para acabar finalmente rellenado el Data WareHouse. Esta tarea también incluirá conseguir que este proceso ETL se ejecute automáticamente, y por tanto que se actualicen los datos del Data WareHouse, cada mediodía. Por lo que el entregable resultante (interno al proyecto) será el Data WareHouse poblado de datos actualizados.

- **P3.2.3. Creación cubo**: Creación de un cubo (entregable interno) a partir del Data WareHouse y configuración de Pentaho para poder realizar desde ahí consultas al cubo.

- **P3.2.4.1 Aplicación web**: Elaboración de una aplicación web cuyo acceso esté restringido y desde la cual el usuario pueda elegir qué cuadros de mando visualizar de entre los que implementemos. Contará también con una página principal, con enlaces a todas las pestañas de la aplicación. Esta misma será un entregable interno al proyecto.

- **P3.2.4.2 Implementación CMI**: Implementación de CMI complejos y a medida de las preferencias del cliente. Se programarán usando HTML, CSS, Java, Json, Ajax, Jquery y Javascript (en particular, Highcharts). Se estima que se realizarán seis y que la creación de cada uno de ellos, entregables internos resultantes de esta tarea, costará unas 16 horas.

- **P3.3 Pruebas**: Incluirá todas las pruebas necesarias para asegurarnos que cumplimos los requisitos y que el funcionamiento es el esperado. Es decir, afianzarnos de que no hemos cometido ningún error. Por tanto, al finalizarla, obtendremos definitivamente el entregable final del producto comentado anteriormente en el apartado 1.1.1.

- **P4.1 Manual**: Redacción de un manual, que será un entregable al cliente y que a su vez será incluido en los anexos de la memoria, que explique cuál es el software a instalar en caso de que desee usar este producto en otro lugar y no en la máquina virtual, detalles de configuración que en ese caso necesite modificar o que deba saber para su uso, etc.

- **P4.2 Memoria**: Elaboración de la memoria del trabajo, que consistirá en la unión de las partes que la conforman realizadas previamente en otras tareas y en la redacción de los apartados restantes que la forman (introducción, implementación, etc). Por tanto, el entregable resultado de esta tarea será esta misma, que será entregado al tutor del proyecto.

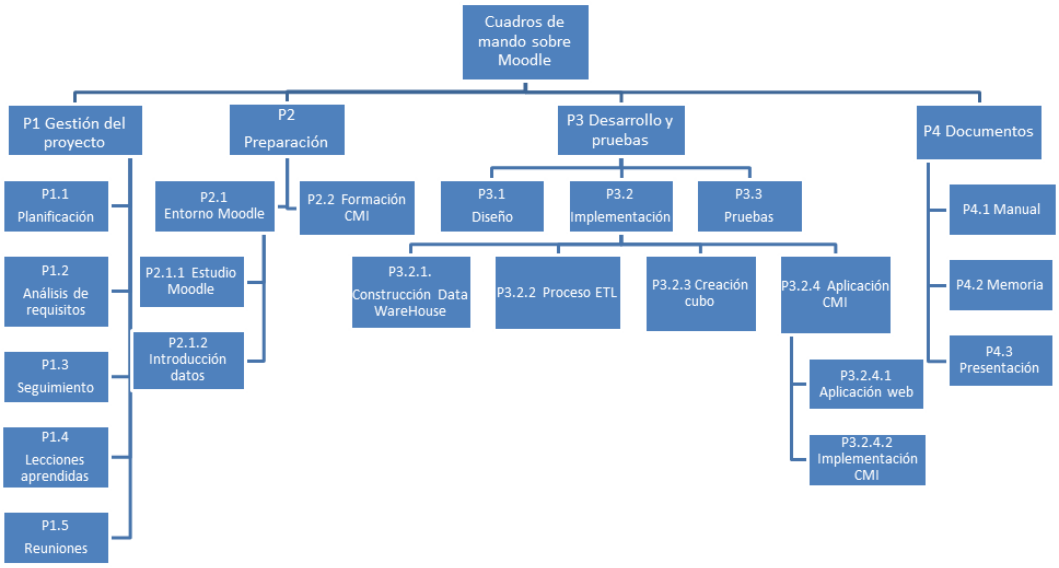


Figura 1.2: Estructura de descomposición de tareas (EDT) del proyecto.

- **P4.3 Presentación**: Realización de la presentación de diapositivas que se utilizará en la defensa de este proyecto o Trabajo Fin de Grado y ensayo de la defensa del mismo. Esta tarea tendrá como entregable resultante dicha presentación.

1.2. Tareas

1.2.1. Orden de realización

Para empezar, realizaremos la planificación del proyecto, teniendo en cuenta lo que nos ha comentado el cliente en la primera reunión en cuanto a definición del alcance del proyecto, objetivos, requisitos, etc. Además, en esta, el cliente nos ha encomendado la misión de investigar Moodle y a raíz de los datos que se almacenan en él, definir una lista de medidas y dimensiones que pueden resultar interesantes. Por lo que al acabar la planificación, pasaremos a la tarea «Estudio Moodle», y seguidamente, a la de «Introducción datos».

Después, vendrá la primera fase de análisis de requisitos. Decimos que es la primera ya que el cliente nos ha pedido que, debido a las limitaciones de tiempo del proyecto, a su imposibilidad de quedar frecuentemente y a las dudas que tenía todavía sobre los requisitos que quería referentes a los CMI, realizáramos las fases de análisis, diseño, implementación y pruebas como si estuviéramos ante dos subproyectos: uno cuyo objetivo es la creación del cubo multidimensional que hemos comentado en otros apartados, y otro destinado a implementar CMI concretos, a los cuales poder acceder mediante una aplicación web. En cada uno de ellos, emplearemos una metodología similar a la de en cascada aunque con solapamientos de implementación y pruebas.

Esta fase de análisis acabará con una redacción de todos los requisitos correspondientes a la primera parte del proyecto, sacados tanto de la primera reunión con el cliente como del análisis

de Moodle que habremos realizado anteriormente. Cuando la acabemos, nos reuniremos con el cliente otra vez para mostrarle dicha lista y que opine sobre ella, terminando al final de la reunión con la lista definitiva de requisitos de la primera parte.

Entonces, pasaremos a la fase de diseño y, una vez la terminemos, comenzaremos con la de implementación. Esta empezará con la tarea de «Construcción Data WareHouse», seguida de la de «Proceso ETL» y acabando con la de «Creación cubo» (sin superponerse en el tiempo dichas tareas). Y mientras se vayan llevando a cabo, se irán realizando todas las pruebas pertinentes.

Tras todas estas fases, habremos llegado ya al fin de la primera parte del proyecto, por lo que nos formaremos en lo necesario para realizar la segunda (tarea «Formación CMI»). Posteriormente, nos reuniremos con el cliente para fijar los requisitos exactos de esta nueva parte y realizar así la fase de análisis. Una vez acabemos la misma, pasaremos a la de diseño y al finalizarla, a la de implementación, en la cual se realizará la tarea «Implementación CMI» alternativamente a la de «Aplicación web». Esta fase será interrumpida en ocasiones por la de pruebas.

Para acabar, elaboraremos el manual de configuración que entregaremos también al cliente.

Notar que a lo largo del proyecto, además de irnos reuniendo con el tutor, iremos realizando un seguimiento, apuntado las lecciones que vayamos aprendiendo y en general, redactando la memoria. Y, una vez haya aceptado el cliente el producto final, nos dedicaremos exclusivamente a terminar correctamente estas tareas y después, a preparar una presentación del proyecto.

1.2.2. Hitos

En las figuras 1.3 y 1.4 se muestran distribuidos, a lo largo del periodo de tiempo en el que se realizará el proyecto, los distintos hitos relevantes del mismo.

Hitos	S1	S2	S3	S4	S5	S6	S7	S8	S9
Reunión inicio proyecto	◆								
Reunión aceptación requisitos parte 1				◆					
Reunión recogida requisitos parte 2							◆		
Reunión aceptación producto									
Depósito proyecto									
Defensa proyecto									

Figura 1.3: (a) Diagrama de hitos (Semanas 1-9, siendo S1 la semana del 31 al 6 de Noviembre).

Hitos	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19
Reunión inicio proyecto										
Reunión aceptación requisitos parte 1										
Reunión recogida requisitos parte 2										
Reunión aceptación producto						◆				
Depósito proyecto								◆		
Defensa proyecto										◆

Figura 1.4: (b) Diagrama de hitos (Semanas 10-19).

Notar que las reuniones a las que nos referimos anteriormente son todas con el cliente.

1.2.3. Estimación de dedicación a cada tarea

El proyecto completo deberá realizarse en 300 horas, de las cuales 285 tendrán que ser de trabajo autónomo y 15 de reuniones. Para llevarlo a cabo con buena calidad, realizamos una estimación del tiempo adecuado a invertir en la realización de cada tarea, como se muestra en el cuadro 1.1.

	Tarea	Tiempo estimado
P1.1	Planificación	20 horas
P1.2	Análisis de requisitos	5 horas
P1.3	Seguimiento	15 horas
P1.4	Lecciones aprendidas	2 horas
P1.5	Reuniones	15 horas
P2.1.1	Estudio Moodle	5 horas
P2.1.2	Introducción datos	5 horas
P2.2	Formación CMI	8 horas
P3.1	Diseño	10 horas
P3.2.1	Construcción Data WareHouse	2 horas
P3.2.2	Proceso ETL	22 horas
P3.2.3	Creación cubo	20 horas
P3.2.4.1	Aplicación web	20 horas
P3.2.4.2	Implementación CMI	96 horas
P3.3	Pruebas	10 horas
P4.1	Manual	2 horas
P4.2	Memoria	30 horas
P4.3	Presentación	13 horas
	TOTAL	300 horas

Cuadro 1.1: Tiempo estimado para la realización de cada tarea.

1.2.4. Periodos de ejecución previstos

Teniendo en cuenta lo comentado a lo largo del apartado 1.2, hemos realizado la repartición en el tiempo de las tareas descritas en la EDT de la forma que se presenta en el diagrama de Gantt expuesto en las figuras 1.5 y 1.6. En él podemos observar los días previstos para la realización de cada una de las tareas del proyecto.

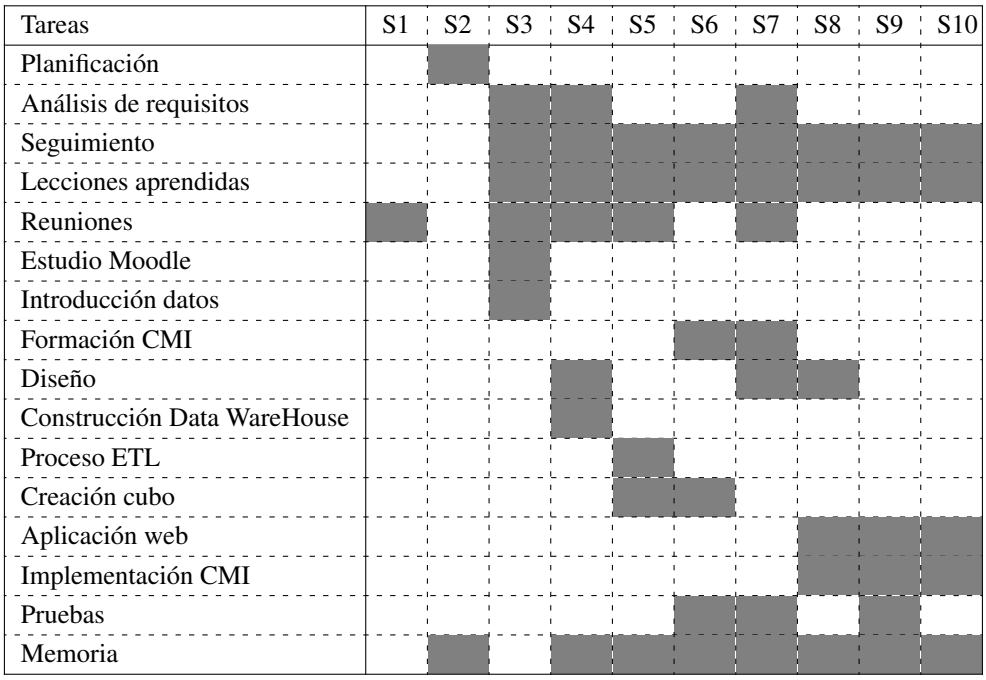


Figura 1.5: (a) Diagrama de Gantt (Semanas 1-10, siendo S1 la semana del 31 al 6 de Noviembre).

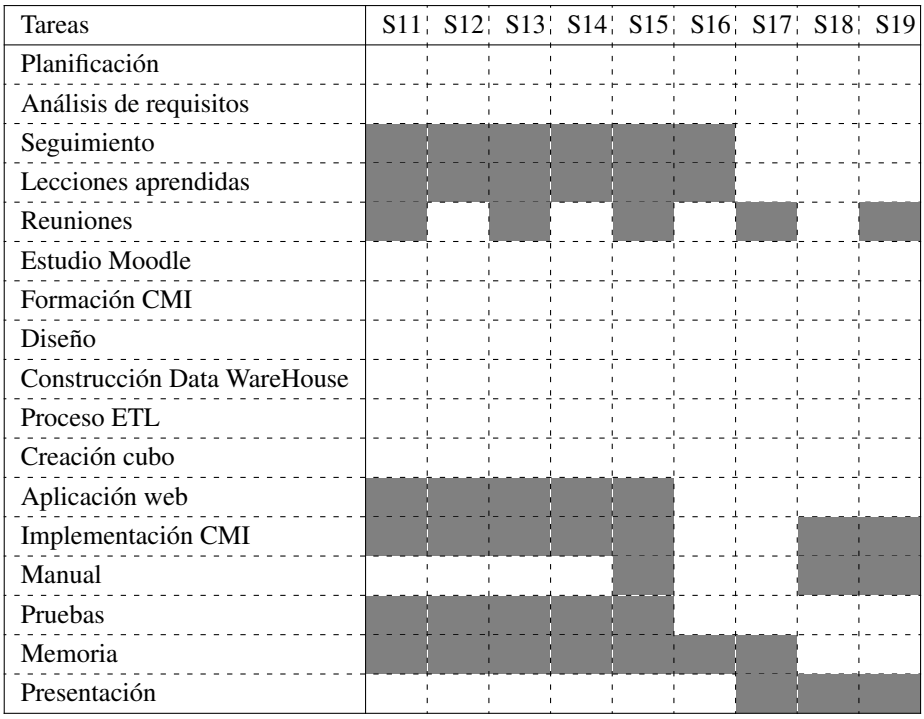


Figura 1.6: (b) Diagrama de Gantt (Semanas 11-19).

1.3. Herramientas de comunicación

Tanto con el cliente como con el tutor del trabajo, la comunicación se dará principalmente a través de correos electrónicos y de reuniones, aunque también es posible que se utilicen llamadas para aspectos puntuales que deban resolverse rápidamente, y enlaces de Google Drive para transferir o compartir archivos.

1.4. Plan de riesgos

En los cuadros 1.2 y 1.3 se detallan los riesgos y las potencialidades respectivamente que hemos previsto que puedan aparecen durante el desarrollo del presente proyecto.

Fuente	Riesgos	Si sucede	Para evitarlo o minimizarlo
Relación con el cliente	Disfunción comunicación	Buscar nuevas vías de comunicación más cercanas a la presencial o en horarias distintos	Preparar las preguntas antes de cada reunión, expresarse con claridad, y acordar con bastante antelación las fechas de las reuniones
Relación con el cliente	Cambios en las especificaciones o alcance	Valorar los cambios y en caso de aceptarlos, planificar las nuevas tareas	Desarrollo abierto que permita fácilmente realizar cambios en el mismo. Contar con margen de tiempo
Tecnología	Presenta limitaciones	Recortar el alcance o buscar alternativas	Analizar antes de elegir herramientas sus posibilidades
Tecnología	Deja de funcionar	Buscar una alternativa y planificar los cambios y actuaciones necesarias	Usar herramientas con soporte y recorrido que sean fiables
Técnico	Se estropea el ordenador, se corrompe la máquina virtual, etc.	Restaurar la copia de seguridad más reciente	Realizar frecuentemente copias de seguridad y almacenarlas ya sea online o en otro dispositivo físico
Datos almacenados en Moodle	Formato inadecuado para realizar un proceso ETL a partir de ellos	Recortar el alcance, de forma que se utilicen solamente los datos que se encuentren en un formato adecuado	Previamente a comenzar el proceso ETL, analizar los datos que se almacenan en Moodle y en qué formato
Recursos Humanos	Enfermedad del responsable del proyecto o mal funcionamiento (avances muy lentos, etc)	Planificar nuevos periodos de tiempo en los cuales se realizarán las tareas o negociar con el cliente nuevas fechas o recortes en el alcance	Tratar de finalizar cada una de las tareas para cuando está previsto en la planificación o incluso anticipar su realización

Cuadro 1.2: Plan de riesgos

Fuente	Potencialidades	Si sucede	Para favorecerlo
Relación con el cliente	Buena sinergia con el cliente	Mantener la actitud y aprovecharla para evitar cambios en el alcance que nos puedan alterar la planificación	Mantener una actitud cordial y asertiva en las reuniones
Tecnología	Facilidades imprevistas con la tecnología	Aprovechar esas facilidades para avanzar y adelantar. Si nos sobrara tiempo se podría mejorar la calidad	Usar tecnologías populares y muy desarrolladas
Recursos Humanos	Funcionamiento extraordinario	Si vamos adelantados a la planificación, mejorar la calidad del producto	Utilizar sistemas que conozcamos y estemos cómodos usando. Realizar lo máximo posible cuando estemos lúcidos

Cuadro 1.3: Plan de potencialidades

1.5. Plan de calidad

A lo largo del proyecto, iremos verificando que cumplimos cada uno de los requisitos indicados en el capítulo 2. Esta actividad, perteneciente a la tarea de «Seguimiento», la realizaremos de forma manual; partiendo de la lista de todos los requisitos impresa, iremos consultándola y marcando en ella aquellos que vayamos cumpliendo para asegurarnos así finalmente que cumplimos todos y que nuestro producto tiene la calidad suficiente que debe tener.

1.6. Plan de cambios

En el cuadro 1.4 detallamos las áreas de cambio valorando el impacto esperado, y quién decidirá si este queda aprobado.

Tipo de cambio	Impacto	Quién decidirá
Plazos	Alto/muy alto	Responsable del proyecto
Uso de otra herramienta BI	Alto, aumentando a medida que avanza el proyecto	Responsable del proyecto
Modificación de medidas, dimensiones o filtros	Medio	Cliente y responsable del proyecto
Cambios en los CMI con respecto a lo planificado	Medio/alto	Cliente y responsable del proyecto
Implementar nuevos CMI	Alto	Responsable del proyecto
Añadir nuevas funcionalidades a la aplicación web	Alto	Responsable del proyecto

Cuadro 1.4: Plan de cambios

Capítulo 2

Análisis de requisitos

2.1. Requisitos parte 1

Los **requisitos funcionales** correspondientes a la primera parte del proyecto (hasta una vez hayamos creado el cubo) son los siguientes.

- Abstracción de los datos, referentes a las siguientes medidas y dimensiones, de la propia base de datos de Moodle en la que están almacenados. Centralización de ellos en otra base de datos (Data WareHouse) tras llevar a cabo el proceso ETL respectivo, siempre y cuando la estructura de Moodle sea la presente en la figura 2.1.
    - Medidas: notas de exámenes online, de tareas, de participación y de exámenes presenciales, calificaciones finales, número de alumnos y número de aprobados y suspensos. Todas aquellas referentes a notas se tratan de sus medias.
    - Dimensiones: año, evaluación, alumno, asignatura, etapa, curso, centro y profesor.

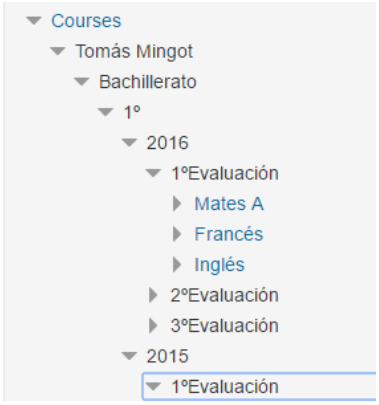


Figura 2.1: Ejemplo del esquema de la estructura de los datos de Moodle (centro ->etapa ->curso ->año ->evaluación ->asignatura).

- Actualización del Data WareHouse automáticamente cada mediodía con respecto a los datos almacenados en la base de datos de Moodle. Solamente se actualizarán si se ha

iniciado sesión en el ordenador pertinente.

- Realización de consultas de alguna de las medidas anteriores en función de una o más de las dimensiones, de forma fiable, sencilla y rápida, sin necesidad de tener conocimientos técnicos de ningún tipo, ni de saber dónde se encuentran almacenados tales datos. Es decir, consultas a un cubo multidimensional que coge los datos del Data WareHouse referentes a las medidas y dimensiones indicadas anteriormente.

- Filtración de los resultados obtenidos en las consultas anteriores. Los filtros posibles de usar serán cualquiera de las dimensiones disponibles en el cubo.

- Posible modificación del Data WareHouse y del cubo, por parte de alguien con los conocimientos técnicos necesarios.

Y los **no funcionales** son los siguientes.

- La herramienta BI desde la cual se realizarán las consultas deberá ser gratuita y fácil de usar (intuitiva, visual).

- El resultado final de esta parte será una máquina virtual Ubuntu 16.04, que conste de todo lo necesario para que se puedan satisfacer todos los requisitos comentados anteriormente.

2.2. Requisitos parte 2

Los **requisitos funcionales** correspondientes a la segunda parte del proyecto son los siguientes.

- Visualización gráfica, a través de cuadros de mando integrales (CMI) realizados a partir de los datos del cubo implementado en la parte 1 del proyecto, de las siguientes relaciones y del siguiente modo. Estos estarán disponibles mediante una aplicación web.

    - Número de alumnos matriculados a lo largo de la historia para cada uno de los centros almacenados en Moodle. Aunque por defecto aparecerá esta información, el usuario dispondrá de la opción de elegir los años, los centros, las etapas, los cursos o las asignaturas para los que desea mostrar el número de alumnos, pudiendo elegir para cada uno de estos factores cero, uno o varios valores. Por ejemplo, si el usuario selecciona las etapas ESO y Bachiller, los centros Tomás Mingot y Jesuitas y el año 2016, mostrará el número de alumnos total, entre los dos centros, que han cursado en 2016 la ESO y el número de los que han realizado Bachiller. En caso de que el usuario decida filtrar los resultados por curso, tendrá que indicar anteriormente a qué etapa se refiere dicho curso.

      Toda esta información será mostrada en un gráfico de sectores y en forma de tabla, la cual deberá contener para cada fila, un acceso directo a una página que contenga una tabla con la lista de los alumnos (junto a sus calificaciones finales) a los que se refiere en cada caso. Además, si el usuario pincha en el gráfico, estará aplicando automáticamente el filtro correspondiente. Por ejemplo, en caso de que pinche en el sector referente a un centro, se actualizarán automáticamente el gráfico y la tabla, mostrando así para cada etapa de ese centro, cuántos alumnos hay. De esta forma, sucesivamente, se podrá ir descendiendo hasta asignatura (año ->centro ->etapa ->curso ->asignatura).

- Número de profesores a lo largo de los años para cada centro almacenado en Moodle. Este CMI deberá ser análogo al comentado en el apartado anterior, pero mostrando el número de profesores en vez de el de alumnos. En este caso, al hacer click en una fila de la tabla, aparecerá en esa misma página una tabla con la lista de profesores a los que se refiere.
- Nota media y número de aprobados y suspensos, para cada uno de los centros almacenados en Moodle. El usuario podrá filtrar esta información análogamente a los cuadros comentados anteriormente, aunque en este caso también lo podrá realizar por evaluaciones. La nota media para las dimensiones escogidas, se mostrará en forma de gráfico y de tabla. En esa misma tabla aparecerá también el número de aprobados y suspensos pero solamente en caso de que se hayan escogido valores para una dimensión (comparación por esa dimensión) o se haya seleccionado máximo una opción de cada una. Al pinchar en cada fila de la tabla, el usuario accederá a una tabla con la lista de alumnos a los que corresponde esa media, junto con su nota media individual para las dimensiones marcadas.

  Además, solamente en caso de disponer de tiempo suficiente, se realizará también lo siguiente. Al pinchar en cada alumno, desplegará su nota en la dimensión siguiente a la última marcada (en forma de gráfico), siendo el orden el siguiente: año ->centro ->etapa ->curso ->evaluación ->asignatura. Por ejemplo, si hay algún curso seleccionado, mostrará para cada evaluación de ese curso, la nota media del alumno. Y si hay alguna asignatura seleccionada (solamente una debe estar marcada en este caso), mostrará para ella las notas de participación, de tareas, de exámenes online y de exámenes presenciales del alumno.

  Por otra parte, también existirá la opción de obtener una comparativa por años del número de aprobados y de suspensos de cada centro. Esta se mostrará en forma de gráfico de doble barra y se podrá filtrar por año de comienzo de la comparación, año final e intervalo que se desea que haya entre los años que se comparan. En caso de no aplicar ningún filtro, la comparación será de todos los años almacenados en Moodle.
- Oferta educativa por centro, es decir, número de etapas impartidas en él. Se podrán filtrar los resultados por año y por centro. Estos datos aparecerán en un gráfico de sectores y en forma de tabla, aunque en la tabla en vez de aparecer el número de etapas, mostrará cuáles son exactamente las que se imparten.

- A la hora de filtrar los resultados de los cuadros de mando, el usuario no podrá elegir una combinación de factores inexistente. Es decir, cada vez que seleccione alguna opción, se actualizarán las opciones posibles del resto de dimensiones de acorde a las ya seleccionadas.
- El usuario podrá descargarse todos los gráficos (en formato pdf) y tablas (en excel), una vez los consulte en la aplicación.
- El usuario solamente podrá hacer uso de los servicios de la aplicación web si se ha autenticado, es decir, si ha introducido correctamente el usuario y la contraseña.
- Toda página perteneciente a la aplicación tendrá un enlace a cada uno de los cuadros de mando realizados y comentados anteriormente, así como un botón para que el usuario cierre su sesión cuando lo desee.
- La aplicación web contará con una página principal desde la cual se podrá acceder a cada uno de los CMI comentados anteriormente. Estos accesos directos deberán ser intuitivos,

de forma que el usuario se pueda hacer una idea del contenido de la página a la que lleva cada uno de ellos.

- Posible modificación de los CMI y de la aplicación en general, por parte de alguien con los conocimientos técnicos necesarios.

Por otra parte, los **no funcionales** son los siguientes.

- El usuario no tendrá que pagar ninguna licencia.
- La aplicación web deberá ser intuitiva y fácil de usar.
- La aplicación web deberá ofrecer una buena navegabilidad.
- El entregable final principal del proyecto al cliente será la máquina virtual resultante de la primera parte del proyecto y que además, permita satisfacer también todos los requisitos correspondientes a la segunda fase del proyecto comentados anteriormente.
- Además, otro entregable al cliente será un manual de configuración que conste de una lista con el software que el usuario deberá instalar en caso de que desee migrar el producto a otro lado, y detalles de configuración a tener en cuenta para desplegar la aplicación, etc.

2.3. Casos de uso

Para aclarar los requisitos, en la siguiente figura se muestra el diagrama de casos de uso referente a este proyecto. Notar que el actor «Usuario» es aquel que ya se ha autenticado.

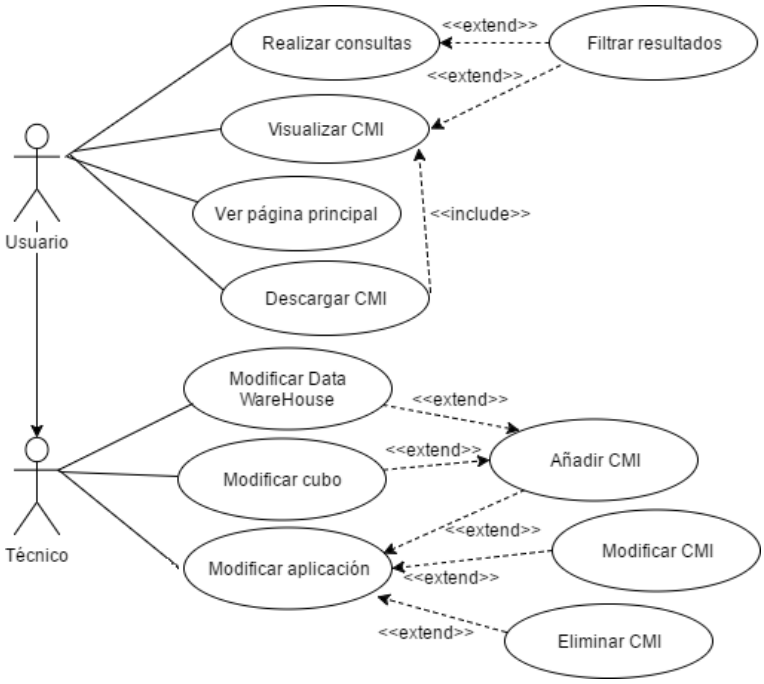


Figura 2.2: Diagrama de casos de uso.

Capítulo 3

Diseño

3.1. Diseño parte 1

Data WareHouse

Toda la información de Moodle se vuelca automáticamente en una base de datos, MySQL en este caso, con 361 tablas, por lo que buscar información en ella resulta una tarea muy laboriosa. Así que para poder extraer más rápidamente y fácilmente la información que resulta de interés al cliente, concentraremos tales datos en otra base de datos a la que llamaremos «datawarehouse».

Una vez creada dicha base de datos MySQL vacía, construiremos su estructura ayudándonos de la herramienta Toad for MySQL. A la hora de crear dicha estructura, seguiremos el modelo básico, conocido como modelo en estrella. Es decir, el Data WareHouse estará formado por tablas de dimensiones (una por cada una de las dimensiones comentadas en el apartado 2.1 y a su vez, cada una tendrá dos columnas: id (clave primaria) y descripción), conectadas a una tabla de hechos central, que tendrá como columnas una por cada una de las medidas que nos interesan analizar, y otra por cada una de las dimensiones que será clave extranjera a la correspondiente tabla de dimensiones. Estas últimas conformarán a su vez la clave primaria de la tabla. En la figura 3.1 se puede observar dicha estructura. En ella, las claves primarias de cada tabla se corresponden con los campos subrayados. Además, todos los campos «id» serán enteros, mientras que los campos «descripcion» serán cadenas de caracteres. Notar también que los campos relacionados con notas, se refieren a las medias correspondientes.

Proceso ETL

Realizaremos el proceso ETL mediante Kettle (Pentaho Data Integration). Crearemos un proyecto por cada una de las tablas del Data WareHouse que deberemos poblar, cuya ejecución implique la extracción de los datos que necesitamos mediante consultas a Moodle (paso «Entrada tabla») y la carga de los mismos en la tabla correspondiente del Data WareHouse (paso «Salida tabla»).

En caso de que se desee rellenar un campo con un valor dependiente de más de un campo devuelto en el resultado de la consulta hecha a Moodle o de operaciones con ellos, necesitaremos un paso intermedio (paso «Valor Java Script modificado») en donde se realicen las operaciones o uniones pertinentes y se declaren y definan las variables necesarias que serán posteriormente asignadas (en el paso «Salida tabla») a los campos respectivos del Data WareHouse.

Finalmente, crearemos una tarea en Kettle que desencadene la ejecución de todos los proyectos creados anteriormente automáticamente cada mediodía, actualizándose así los datos del Data

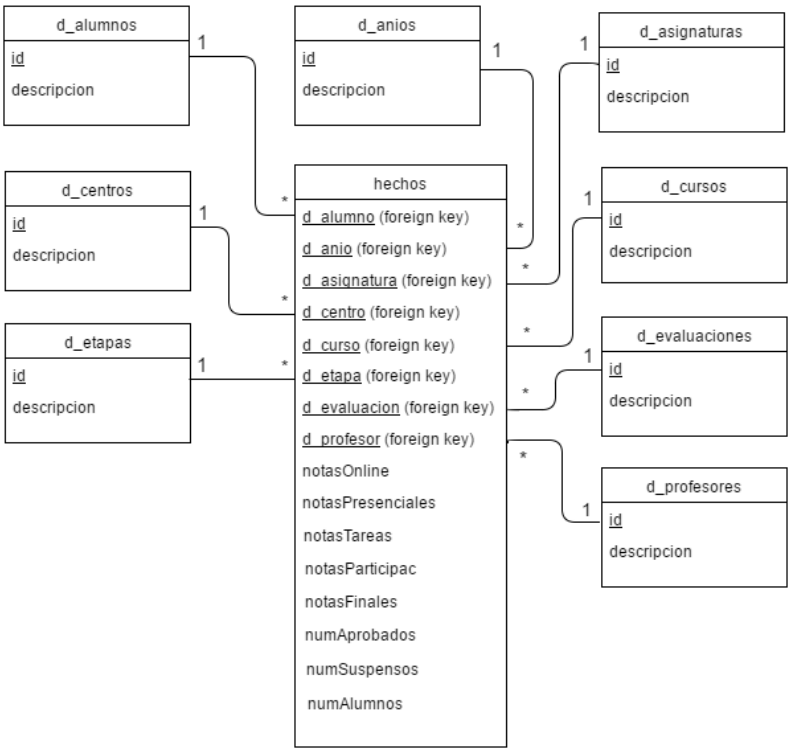


Figura 3.1: Estructura del Data WareHouse (modelo en estrella).

WareHouse.

Una vez concluido el proceso ETL, ya dispondremos de un Data Warehouse poblado de toda la información que interesa al cliente. Por lo que ahora tendremos que dedicarnos en conseguir realmente aprovechar esta información, optimizando la toma de decisiones basadas en ella.

Cubo

Mientras que el Data WareHouse almacena la información tal y como la hemos obtenido de la base de datos de Moodle, los sistemas OLAP hacen agregaciones, sumarizaciones u otras operaciones a esos datos, y los organizan en cubos o almacenamientos especiales, independientes del sistema transaccional, para permitir una rápida recuperación ante una consulta.

En nuestro caso, crearemos un cubo OLAP Mondrian, que nos permitirá realizar distintas combinaciones de sus elementos para visualizar los resultados desde diversas perspectivas. El diseño del cubo (matriz multidimensional) será el típico de un cubo: sus ejes serán las dimensiones (tablas de dimensiones definidas en el Data WareHouse) y los valores presentes en la matriz serán los correspondientes a las medidas, es decir, valores precalculados a partir de los contenidos en la tabla de hechos del Data WareHouse. Lo realizaremos mediante la herramienta Schema WorkBench.

Consultas

Las consultas serán realizadas gratuitamente mediante Pentaho, en particular utilizando el plugin Saiku Analytics. Por lo que seguirán el siguiente modelo (ver figura 3.2), el cual es

muy visual y sencillo de usar. Bastará con indicar cuáles deseamos que sean las medidas, las dimensiones y los filtros, de entre las posibilidades que tenemos en el cubo.

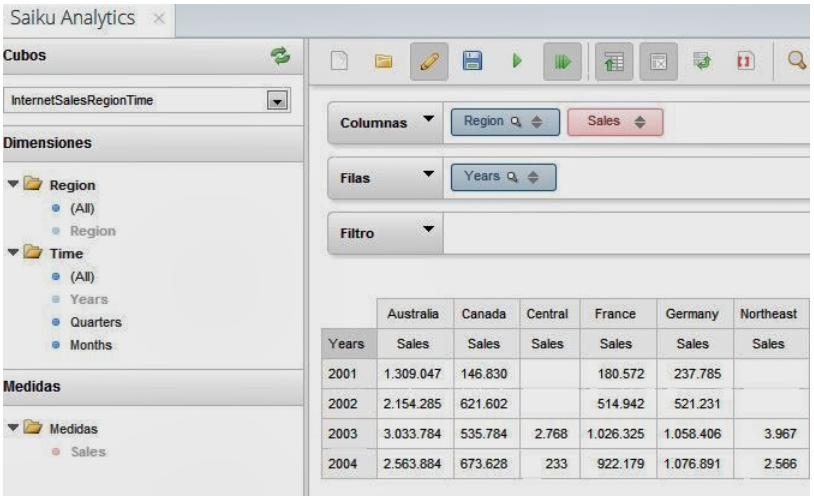


Figura 3.2: Ejemplo formato consulta usando Saiku Analytics en Pentaho (en este caso, ventas en función del país y del año).

Pruebas

Las pruebas que haremos para cerciorarnos de que hemos completado la primera parte adecuadamente serán las siguientes.

- Realizaremos todo tipo de consultas desde Pentaho, seleccionado diferentes medidas, dimensiones y filtros, asegurándonos que devuelven los resultados que esperamos.
- Introduciremos nuevos datos en Moodle, modificaremos y eliminaremos algunos ya existentes (de los que se utilizan en el proceso ETL), e iremos comprobando si realmente el Data WareHouse es poblado con la información que le corresponde y en el formato adecuado, y si tanto él como el resultado de las consultas son actualizados adecuadamente.

3.2. Diseño parte 2

A pesar de no ser «Visualizar CMI» una actividad compleja, debido a la importancia que tiene con respecto a este proyecto, hemos decidido mostrar en la figura 3.3, a través de un diagrama de actividad, cómo discurrirá realmente.

Interfaz

La aplicación estará estructurada en las siguientes pestañas, a las cuales habrá mínimo un acceso directo desde toda página de la aplicación:

- Inicio: En caso de que el usuario haya iniciado ya sesión, esta tendrá el aspecto presente en la figura 3.4. Mientras que en caso contrario, esta será una página de inicio a la aplicación.

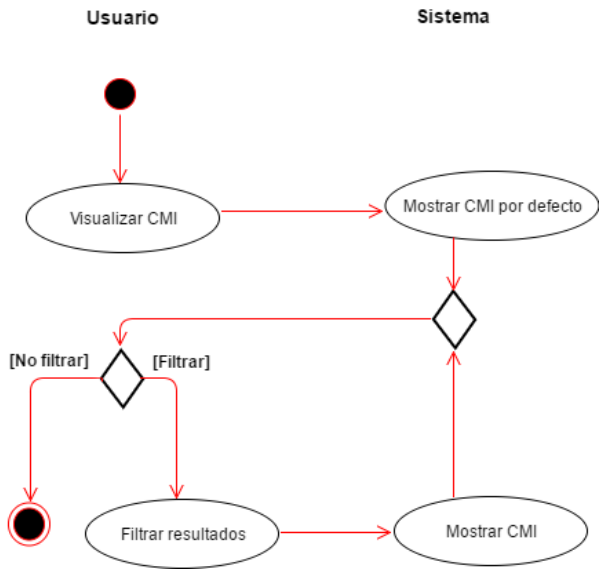


Figura 3.3: Diagrama de actividad referente a «Visualizar CMI».

- Alumnos. En la figura 3.5 podemos observar un boceto de ella.
- Profesores. Tendrá un aspecto semejante a la de alumnos, con la diferencia que al hacer click en una fila de la tabla aparecerá otra tabla justamente debajo con los nombres de los profesores.
- Evaluación. Su interfaz será similar a la presente en la figura 3.6.
- Oferta. Presentará un aspecto parecido a la de alumnos pero solamente con los filtros año y centro.

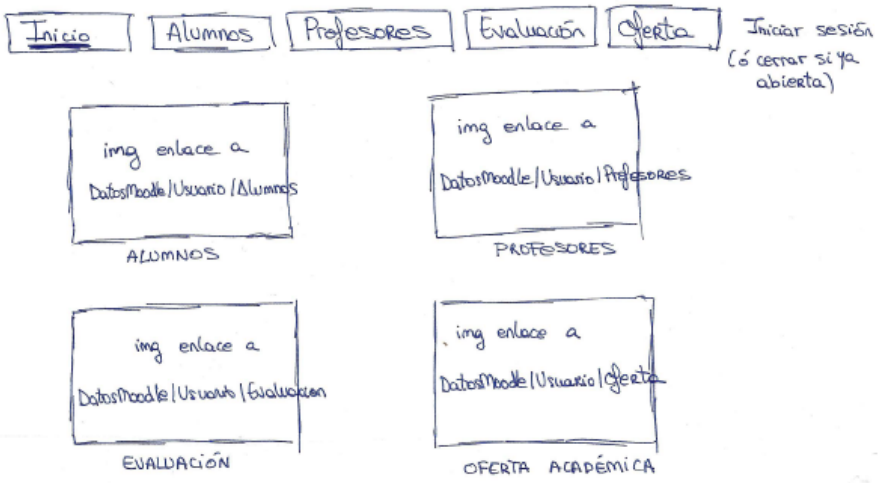


Figura 3.4: Página de inicio una vez el usuario haya iniciado sesión (DatosMoodle/Usuario).

Figura 3.5: Cuadro de mando referente al número de alumnos matriculados (página DatosMoodle/Usuario/Alumnos).

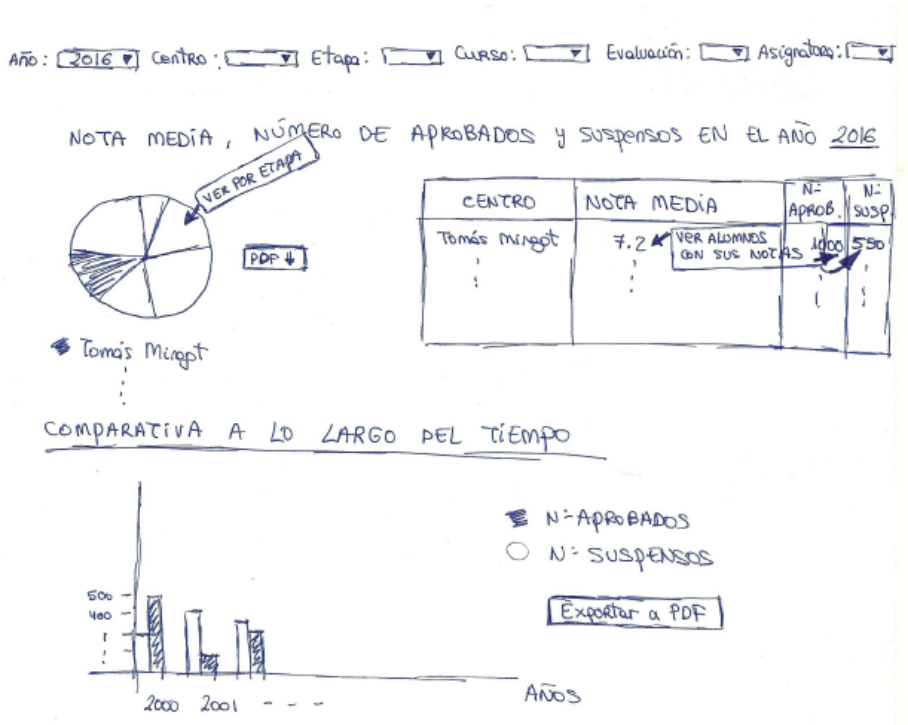


Figura 3.6: Cuadro de mando referente a la nota media, número de aprobados y de suspensos (página DatosMoodle/Usuario/Evaluación).

Tecnologías

La aplicación será realizada mediante NetBeans y utilizando los siguientes lenguajes de programación: HTML, CSS, Java, JavaScript, JQuery, Json y Ajax. En particular, para la creación

de los gráficos utilizaremos la librería Highcharts de JavaScript, mientras que para las tablas usaremos la librería Google Visualization. Por otra parte, para extraer los datos que necesitemos del cubo, al ser este una base de datos multidimensional, tendremos que indicar la consulta MDX correspondiente en cada caso. Afortunadamente no tendremos que investigar nosotros su sintaxis ya que al realizar una consulta con la interfaz de Pentaho, este mismo nos ofrece la opción de ver la consulta MDX que realiza al cubo internamente, por lo que bastará con que copiemos dicho código.

Para la realización de nuestra aplicación deberemos implementar para cada una de las pestañas comentadas anteriormente, así como para la muestra de la lista de alumnos correspondientes con sus notas medias, una vista o jsp y un servlet. Además de una pareja de ellos destinado a controlar el acceso a la aplicación, otra para la página de inicio, y un servlet que se encargue de cerrar la sesión. Por otra parte, ya que los CMI serán actualizados dinámicamente mediante Ajax según el filtro que se aplique, será necesario crear otro servlet para cada CMI. Este será llamado desde la petición Ajax realizada en el respectivo jsp y le devolverá la información necesaria para crear tanto el gráfico como la tabla respectiva. Además, deberemos crear otro servlet que gestione la petición Ajax producida al hacer click sobre una fila de la tabla que muestra el número de profesores y otro que se encargue de devolver lo necesario para realizar el gráfico referente a la comparativa de número de aprobados y suspensos de cada centro. A su vez, como las diferentes opciones posibles a marcar en los filtros se irán actualizando a medida que se marquen opciones en alguno de los restos de los filtros, es decir, mediante peticiones Ajax, también tendremos que realizar un servlet para cada uno de los filtros.

Cabe destacar también que en nuestra aplicación las vistas o jsp se encargarán exclusivamente de su visualización, siendo en los servlets donde se encuentre la lógica de negocio. Además en las vistas utilizaremos EL, complementándolo con JSTL para implementar la lógica de las mismas (iterar colecciones, etc).

Pruebas

Las pruebas que llevaremos a cabo para asegurarnos de que hemos finalizado esta segunda parte del proyecto con éxito serán las siguientes.

- Comprobaremos que no se puede elegir ninguna combinación inexistente de opciones a la hora de filtrar los resultados de los cuadros.
- Para cada cuadro de mando realizado, haremos todo tipo de combinaciones con los filtros disponibles y nos aseguraremos que los resultados son los esperados en cada momento. Así como que funcionan correctamente todos los enlaces y los eventos programados.
- Descargaremos todos los gráficos y tablas, comprobando que la descarga se realiza adecuadamente.
- Introduciremos nuevos datos en Moodle y consultaremos otra vez los cuadros de mando asegurándonos de que devuelve los resultados actualizados.

Capítulo 4

Implementación

En este capítulo comentaremos ordenadamente los distintos pasos que hemos ido realizando en la fase de implementación del proyecto, las desviaciones producidas respecto de lo diseñado, los problemas o dificultades que nos han ido surgiendo y cómo las hemos conseguido solventar o resolver, etc.

4.1. Creación Data WareHouse

Creamos una base de datos vacía de nombre «datawarehouse» y posteriormente, usamos la herramienta Toad for MySQL Community para establecer una conexión al «datawarehouse» y para dotarle de la estructura que le corresponde siguiendo lo comentado en el apartado 3.1 (ver figura 4.1).

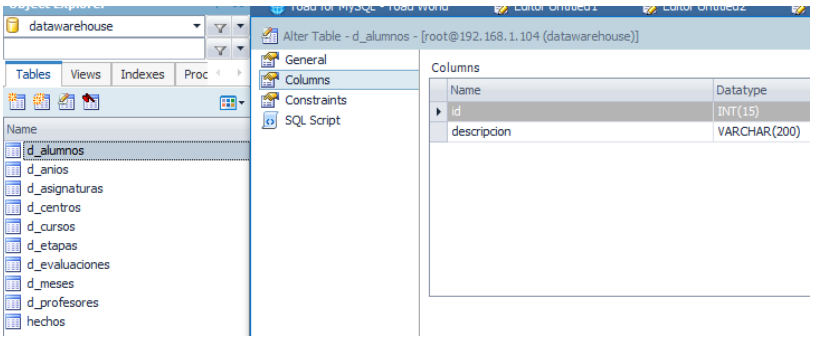


Figura 4.1: Creando la estructura del Data Warehouse (modelo en estrella), mediante Toad for MySQL Community.

En un principio, aparentemente parecía que habíamos conseguido completar este paso con éxito y facilidad, pero al día siguiente nos surgió el siguiente problema. Al ir a iniciar sesión en la máquina, introducíamos correctamente el usuario y la contraseña y no nos permitía iniciar sesión. Finalmente, tras buscar información sobre ello, averigüé que era debido a que habíamos agotado la memoria en el disco duro de la máquina virtual.

Afortunadamente, había hecho instantáneas de la máquina muy recientemente, por lo que partí de la más reciente y cambié la ubicación del directorio donde se almacenaban todos los

datos relacionados con MySQL del disco duro a un disco montado con el administrador de discos del propio Ubuntu. Para ello, fui siguiendo todos los pasos presentados en [1]. Para prevenir que esto volviese a ocurrir, realicé lo mismo con el directorio de Moodle. Aunque en este caso tuve que crear después un enlace simbólico a la anterior ubicación donde se encontraba.

4.2. Proceso ETL

Lo primero que hacemos es decidir los datos de Moodle exactos con los que rellenaremos las tablas del Data WareHouse (recordar su estructura mostrada en la figura 3.1), averiguar la localización de los mismos en la base de datos de Moodle y estudiar cómo se puede llegar a ellos mediante consultas MySQL.

Realizamos esta tarea teniendo en cuenta que la estructura de Moodle es la comentada en el apartado 2.1 (figura 2.1), y debe seguirse la misma en caso de añadir nuevos cursos, centros, años... para que los datos en el Data WareHouse se actualicen correctamente. Cabe destacar que aunque nosotros consideramos que las etapas son ESO, Bachillerato, Primaria..., se podría incluir por ejemplo en esta fase los distintos tipos de Bachillerato que hay. Otra opción para conseguir realizar esta distinción si así se desea o si se quiere saber las diferencias existentes entre unas clases u otras del mismo curso, sería simplemente referirnos a cada uno de los cursos por el número del curso y la letra de la clase a la que queremos hacer referencia (1ºA, 1ºB, etc).

Por una parte, en el cuadro 4.1 indicamos los datos que utilizaremos para poblar las tablas de dimensiones, junto a su procedencia dentro del modelo de Moodle. Dicha procedencia se ha aprendido en la tarea «Estudio Moodle» realizada ayudándonos de Toad. Además, también incluye los datos exactos con los que preveemos que las rellenaremos. A diferencia de lo establecido en la fase de diseño, los campos «id» no serán en todos los casos de tipo entero, por lo que modificamos el tipo de los mismos mediante Toad.

Por otra parte, poblaremos la tabla de hechos de la siguiente manera. Para rellenar sus columnas que son clave extranjera a cada una de las tablas de dimensiones, necesitaremos unir, entre otras, todas las tablas de Moodle utilizadas para rellenar las de dimensiones. Para aclararnos, tras investigar la base de datos de Moodle, hemos realizado el siguiente esquema (ver figura 4.2).

Para rellenar cada una de las columnas de la tabla de hechos correspondientes a las medidas y que están relacionadas con notas, es decir, notasOnline, notasPresenciales, notasTareas, notasParticipac y notasFinales, tendremos que unir la tabla mdl_role_assignment usada anteriormente referida a los alumnos, con una instancia de la tabla mdl_grade_grades. Se unirán por los campos userid de ambas. Y posteriormente, esta última se unirá con mdl_grade_items por los campos itemid e id respectivamente. Se distinguirán unas notas u otras por el valor de la columna idnumber (tareas, [[expresencial]], examenes o participacion) de las tablas mdl_grade_items usadas, pero para todas ellas el valor de la columna courseid deberá coincidir con el de la columna id de la tabla mdl_course anteriormente usada.

El resto de columnas las rellenaremos de la siguiente forma: numAprobados valdrá 1 si notasFinales tiene un valor mayor o igual que 5, y 0 en caso contrario; numSuspensos valdrá 0 si numAprobados vale 1, y 1 en el caso opuesto. Por último numAlumnos valdrá siempre 1, ya que por cada uno de los alumnos matriculados en un centro, etapa, año, evaluación y asignatura habrá solamente una fila en la tabla de hechos. Para realizar todas estas transformaciones, fuimos siguiendo los esquemas que realizamos en la fase de diseño y lo redactado en el apartado 3.1 referente al proceso ETL, aunque tuvimos que realizar también ciertos cambios o acciones que no habíamos previsto.

Moodle (tablas y campos a usar)	**Data WareHouse** (tabla a llenar y equivalencias de sus campos con los de Moodle)
mdl_user (id, firstname, lastname, deleted) y mdl_role_assignments (roleid, userid), con id = userid	d_alumnos (Si deleted!=1 y roleid = 5: id = id; descripcion = firstname + lastname)
mdl_user (id, firstname, lastname, deleted) y mdl_role_assignments (roleid, userid), con id = userid	d_profesores (Si deleted!=1 y roleid = 3: id = id; descripcion = firstname + lastname)
mdl_course_categories (name, depth)	d_anios (Si depth=4: id = name; descripcion = name)
mdl_course_categories (name, depth)	d_evaluaciones (Si depth=5: id = name[0]; descripcion = name + 'intervalo de fechas de cada evaluación')
mdl_course (shortname, fullname, category)	d_asignaturas (Si category !=0: id = shortname; descripcion = fullname)
mdl_course_categories (id, name, depth)	d_centros (Si depth=1: id = id; descripcion = name)
mdl_course_categories (name, depth)	d_cursos (Si depth=3: id = name; descripcion = name + 'curso de la etapa correspondiente')
mdl_course_categories (name, depth)	d_etapas (Si depth=2: id = name; descripcion = name)

Cuadro 4.1: Relación entre la base de datos de Moodle y las tablas de dimensiones del Data WareHouse que rellenaremos.

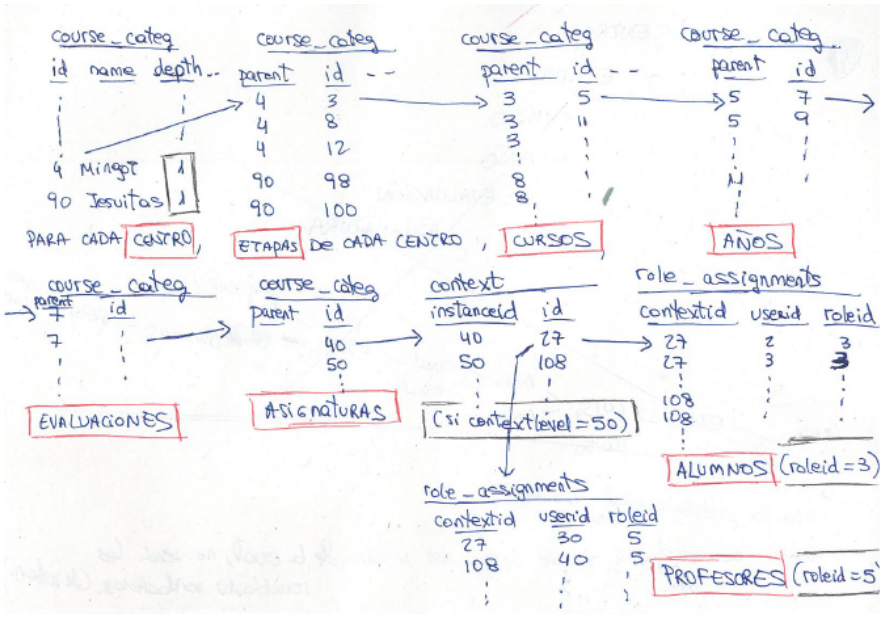


Figura 4.2: Esquema relación tablas de Moodle usadas para poblar las columnas de la tabla de hechos que son clave extranjera de cada una de las tablas de dimensiones.

Una vez tenemos todo planificado, procedemos a realizar las transformaciones. Tal y como diseñamos en el apartado 3.1, creamos diferentes proyectos en Kettle y, en cada uno de ellos, ponemos un paso «Entrada tabla» con el cual obtenemos, mediante una consulta MySQL, los datos que necesitamos para esa transformación de las tablas de Moodle (ver figura 4.3), y otro «Salida tabla» para cargar la información oportuna en la tabla y campos adecuados del Data WareHouse (ver figura 4.4). Para ello, es necesario previamente haber configurado las conexiones a las dos bases de datos. Un fallo típico a lo largo del desarrollo de este proceso ha sido olvidar reconfigurarlas cada vez que cambiábamos de dirección ip, no funcionando entonces los proyectos de Kettle que habíamos creado anteriormente.

Además, colocamos en medio de los dos pasos citados, otro denominado «Valor Java Script modificado» (ver figura 4.5), en aquellos proyectos en los que algún valor de los campos que vamos a poblar no se obtenga directamente de los campos de Moodle obtenidos en una consulta.

A lo largo de este proceso, han ido surgiendo problemas que hemos tenido que ir resolviendo. Por ejemplo, al ejecutar los proyectos de Kettle, obteníamos un error ya que trataba de insertar los datos que ya habían sido insertados en anteriores ocasiones y por tanto, cuyas claves primarias ya estaban en el Data WareHouse. Este problema se podía solucionar simplemente con marcar en los pasos «Salida tabla» la opción de que se vaciase en cada caso la tabla correspondiente del Data WareHouse, previamente a realizar la inserción de los datos. Sin embargo, consideramos que esta era una mala práctica ya que resultaría muy ineficaz en caso de que algún día se tuviesen muchos datos. Por lo que finalmente nos decantamos por añadir, en cada uno de los proyectos de Kettle, que si al tratar de poblar la tabla surgía dicho error, se ejecutase un script SQL que actualizase las filas de dicha tabla con los valores nuevos (ver figura 4.6). Y si esta última orden fallaba, escribiese los campos que trataba de insertar en un fichero de texto, con la finalidad de percatarnos del error (ver figura 4.7).

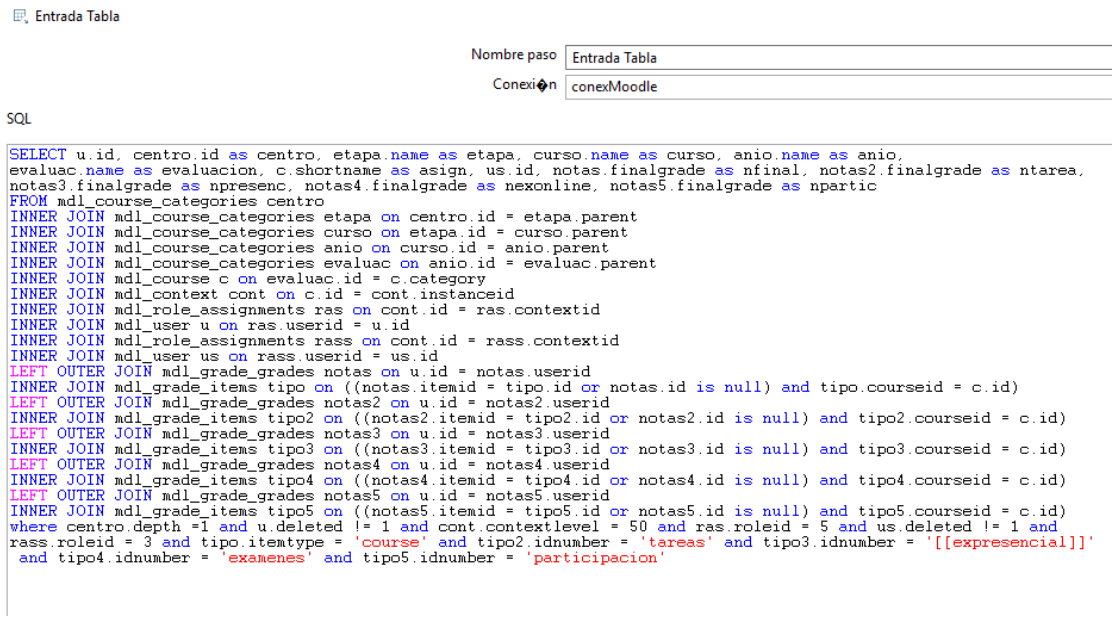


Figura 4.3: Extrayendo, con Kettle, los datos necesarios de Moodle para poblar la tabla de hechos.

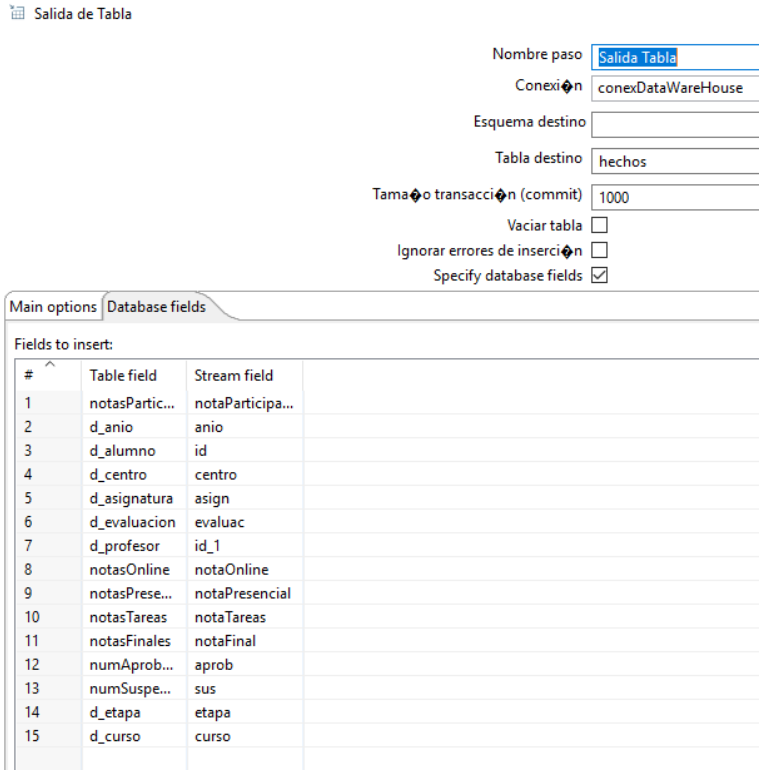


Figura 4.4: Poblando, con Kettle, la tabla de hechos del Data WareHouse.

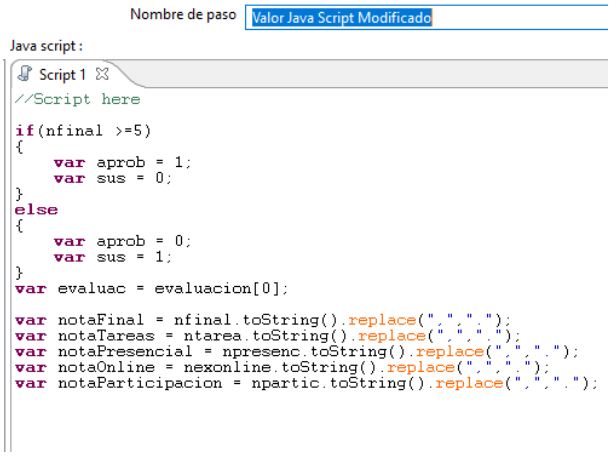


Figura 4.5: Obteniendo, con Kettle, el valor de algunas de las variables con las que rellenaré la tabla de hechos.

Al tratar de realizar el script SQL que actualizase la tabla de hechos (ver figura 4.6), fallaba debido a lo siguiente. El tipo de los campos referentes a las notas obtenidos de Moodle era decimal(10,5), por lo que las calificaciones se guardaban impresas con comas. De forma que al tratar de ejecutar la consulta MySQl, en el que las actualizábamos con los nuevos valores

almacenados en Moodle, se encontraba con las comas y, al tener estas ya un significado en la sintaxis de MySQL, fallaba. Para resolverlo, lo que hicimos fue, en el paso «Valor Java Script modificado», crear nuevas variables resultantes de reemplazar las comas por puntos en cada uno de esos campos, tal y como aparece en la figura 4.5, y modificar el paso «Salida tabla» para asociar las nuevas variables a los campos correspondientes del Data WareHouse. Tras hacer esto, seguía fallando debido a que habíamos definido esos campos en el Data WareHouse como decimal(10,5), por lo que automáticamente guardaba las notas con comas, a pesar de haberlas pasado con puntos. Finalmente, lo solucionamos modificando el tipo de los campos relacionados con notas en el Data WareHouse. Los pusimos como varchar.

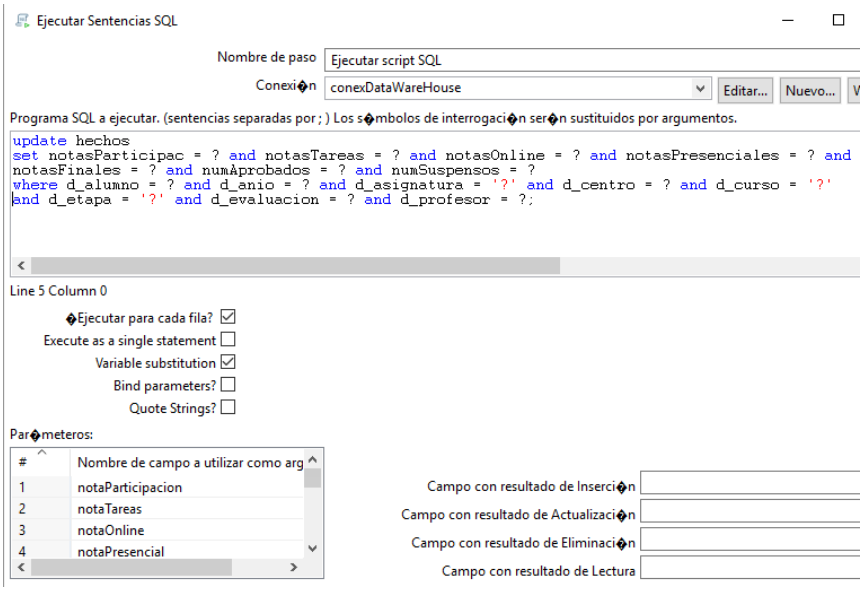


Figura 4.6: Script SQL que actualiza las filas de la tabla de hechos.

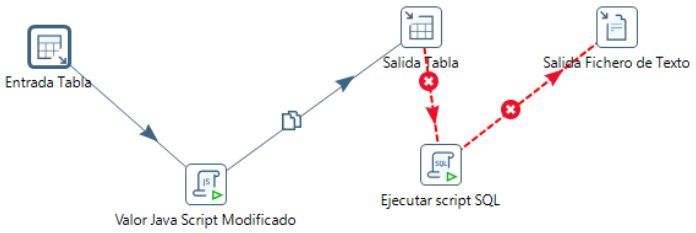


Figura 4.7: Ejemplo esquema transformación Kettle.

A medida que íbamos realizando todo tipo de pruebas para asegurarnos de hacer adecuadamente el proceso ETL, descubríamos más errores o aspectos que no habíamos planeado o que debíamos hacer de forma diferente a como habíamos planeado. Algunos de ellos son: debíamos hacer distincts en las consultas MySQL para evitar repeticiones; tuvimos que ampliar mediante

Toad la longitud de algunos campos de tipo varchar, de las tablas del Data WareHouse, ya que no podíamos volcar algunos datos de Moodle pues excedían la longitud máxima que les habíamos otorgado; en la consulta a Moodle para poblar la tabla de hechos necesitábamos unir también las dos tablas mdl_role_assignments con dos tablas mdl_user, ya que debíamos imponer la condición de que tales usuarios (alumnos y profesor) no hubiesen sido eliminados de la base de datos; la identificación de las notas finales se hacía por el campo itemtype de la tabla mdl_grade_items y debía de valer 'course'; los join que unían los alumnos con sus distintos tipos de notas (tablas mdl_grade_grades) debían ser left outer ya que sino solamente guardábamos en la tabla de hechos los datos referentes a alumnos matriculados a los que ya se les había establecido alguna nota, no pudiendo consultar así al inicio de la evaluación el número de alumnos matriculados... Notar que para conseguir esto último, no fue solamente suficiente el uso de los outer join, ya que el funcionamiento de estos era bloqueado al necesitarse la unión de las tablas mdl_grade_grades y mdl_grade_items. Finalmente, tras darle muchas vueltas y probar de otras maneras sin éxito (debido principalmente a que las claúsulas relacionadas con estas tablas, presentes en el where, fallaban si valía null) decidí añadir «or notas.id is null» en la condición de cada una de las uniones de las tablas mdl_grade_grades y mdl_grade_items. Todo esto se entiende mejor observando la figura 4.3.

Para realizar todas las pruebas, en ocasiones truncábamos tablas con Toad para posteriormente volverlas a poblar y comprobar si ya se llenaban como correspondían. Al hacer esto, las claves extranjeras definidas en la tabla de hechos fallaban e incluso se borraban automáticamente. Por lo que finalmente decidimos realizar todos los pasos del proceso ETL sin tener estas claves definidas, y volverlas a crear una vez habíamos acabado supuestamente, comprobando después que todo seguía funcionando sin errores.

Con respecto a automatizar la actualización de los datos del Data WareHouse, creamos una tarea en Kettle que se encargase de ejecutar las transformaciones realizadas anteriormente, es decir, el proceso ETL (ver figura 4.8).

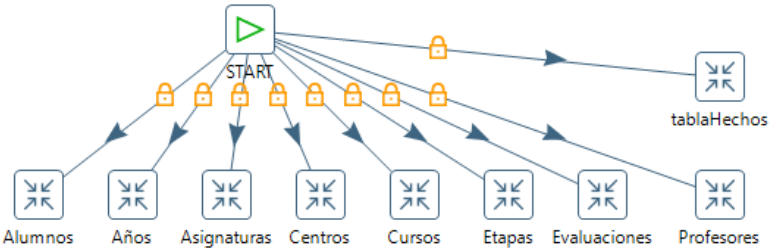


Figura 4.8: Tarea Kettle que desencadena el proceso ETL completo.

Pero, a diferencia de lo diseñado, no programamos con Kettle que se ejecutase cada mediodía, ya que para que así ocurriese, debía estar iniciada la herramienta, lo cual nos resultaba molesto. Por lo que para ello, lo que hicimos es crear un ejecutable que desencadenase la ejecución de la tarea programada en Kettle (ver figura 4.9). Y, a continuación, tras barajar también la posibilidad de crear un acceso directo a dicho ejecutable, decidimos programar, mediante el programador de tareas de Windows 10, que se ejecutase tal ejecutable cada mediodía (ver figura 4.10), en caso de que la sesión y la máquina virtual (y en ella, el servicio mysql) estuviesen iniciadas. Al tratar de hacer esto último, nos surgió la incidencia de que esta no se ejecutaba. Tras recrearla, nos percatamos finalmente que el motivo era que automáticamente, al entrar a propiedades de la

tarea, se ponía que al ejecutarla, se usase la cuenta de usuario Laura, la cual debía coincidir con el autor, que era LAURA\laura.

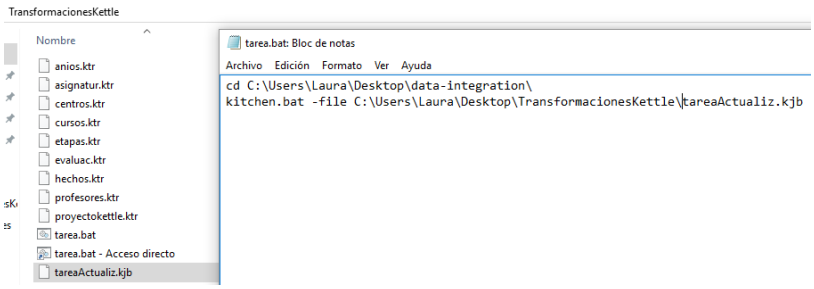


Figura 4.9: Contenido del ejecutable correspondiente a la tarea de Kettle.

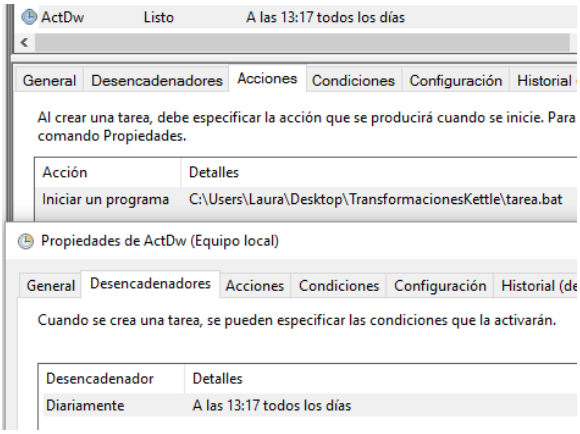


Figura 4.10: Programación de la ejecución de la tarea automáticamente, con el programador de tareas de Windows 10.

Notar que cuando vayamos a realizar la entrega del producto al cliente, le entregaremos a este tanto la tarea de Kettle como el ejecutable, para que pueda optar por programar la ejecución automática de la misma con el programador de tareas que desee (el propio de Windows 10, uno de Ubuntu en la propia máquina virtual, etc).

4.3. Cubo

En esta sección se explica la creación de un cubo OLAP Mondrian a través de una herramienta de interfaz denominada «Schema Workbench», como dijimos en el apartado 3.1.

Cada uno de los valores que contienen esta matriz multidimensional, serán hallados a partir de una operación sobre los datos presentes en la columna de la tabla de hechos, referente a la medida a la que corresponde ese valor. Si esta es alguna de las relacionadas con calificaciones, la operación será la media y si es el número de aprobados o suspensos será la suma. Cabe destacar que con la suma lo realizamos también inicialmente para el número de alumnos pero finalmente, tras percatarnos en la fase de pruebas que lo habíamos hecho erróneamente ya que un mismo alumno aparece varias veces en la tabla de hechos, llegamos a la conclusión de que la operación

era un «distinct count» sobre la columna id de la tabla d_alumnos. De esta forma, pudimos también borrar de la tabla de hechos del Data WareHouse la columna numAlumnos.

Teniendo en cuenta lo comentado hasta ahora, veamos los pasos que hemos ido siguiendo para la creación del cubo. Lo primero que realizamos es establecer una conexión al Data WareHouse desde Workbench (Options ->Connection). Una vez la tenemos, creamos un «schema». Dentro de él, un cubo, y en este, una tabla (referente a la tabla de hechos), todas las dimensiones y todas las medidas (ver figura 4.11).

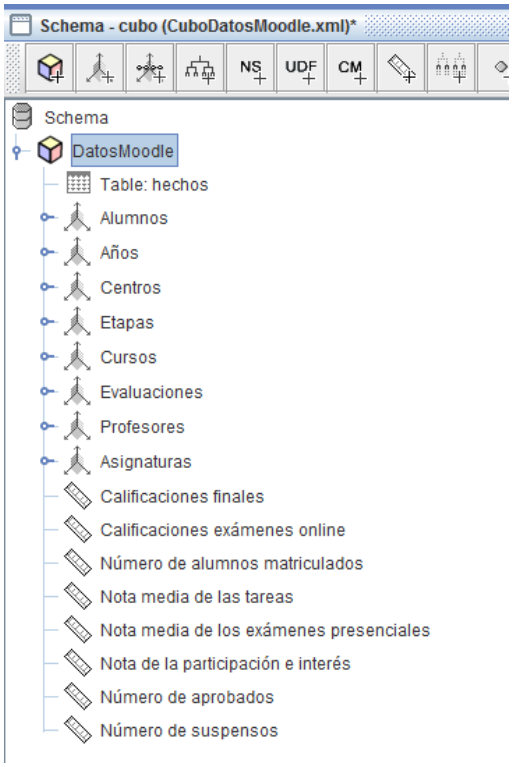


Figura 4.11: Esquema cubo (creado con Workbench).

Posteriormente, dentro de cada una de las dimensiones, creamos una «hierarchy», indicando la clave primaria de la tabla de dimensiones correspondiente del Data WareHouse. Y dentro de esta, un «level» y una tabla con el mismo nombre que la tabla de dimensiones respectiva. Y, para cada medida, indicamos en la fila «aggregator» la operación a realizar con los datos de la columna «column». En la figura 4.12 podemos observar esto para la medida calificaciones de exámenes online.

Attribute	
name	Calificaciones exámenes online
description	Calificación media de los exámenes online realizados
aggregator	avg
column	notasOnline

Figura 4.12: Configurando en el cubo la medida: calificaciones de exámenes online.

4.4. Consultas

Lo primero que debemos hacer es crear un «datasource» en Pentaho, es decir, configurar la conexión al Data WareHouse, e importar el cubo realizado que consultaremos (ver figura 4.13).

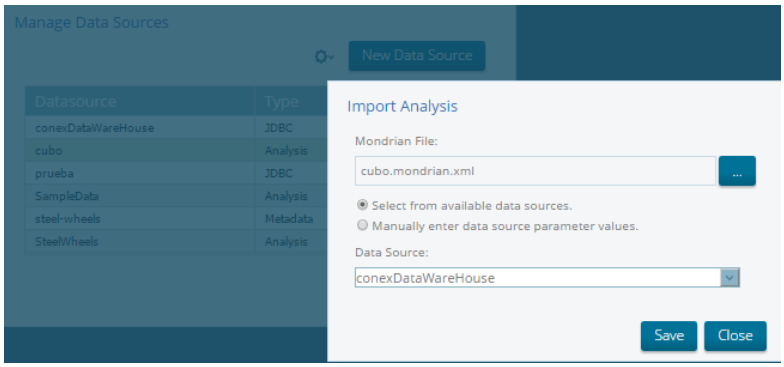


Figura 4.13: Importamos el cubo que consultaremos (Pentaho).

Una vez realizados todos los pasos anteriores, ya podemos realizar consultas al cubo desde Pentaho (Create new ->Saiku Analytics ->Create a new query ->Seleccionar el cubo que deseamos consultar) (ver figura 4.15). Y así llegamos al fin de la primera fase del proyecto.

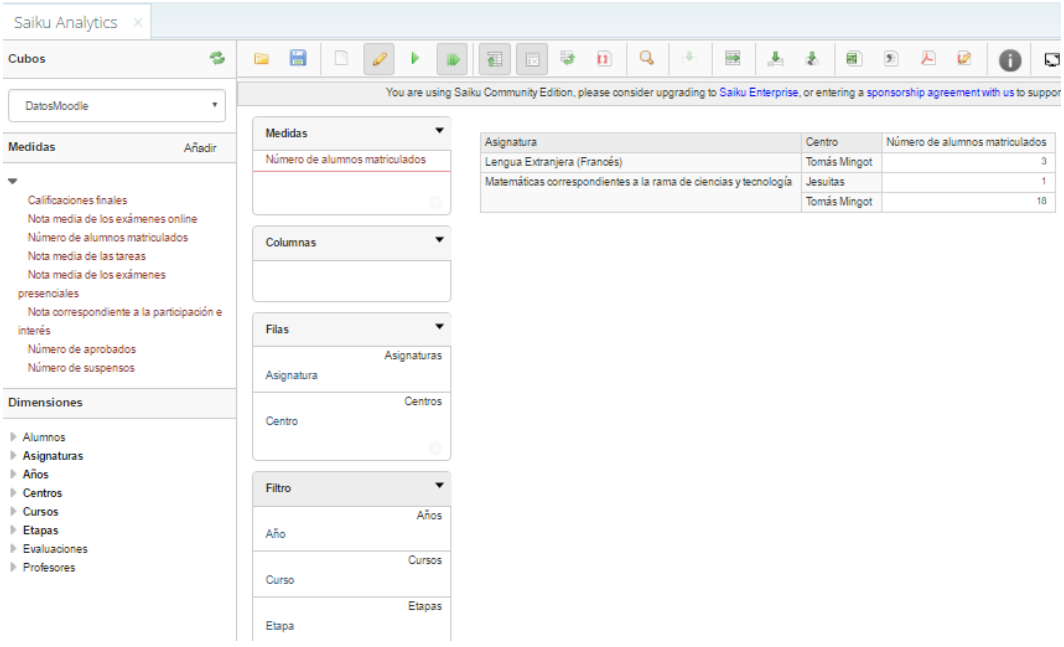


Figura 4.14: Ejemplo consulta al cubo desde Pentaho.

Notar que tras reunirnos con el cliente para presentarle todo esto y pasar así ya a la segunda fase del proyecto, este nos pidió cuadros de mando relacionados con el número de profesores,

lo cual no nos había comentado previamente y por tanto no habíamos planeado ni realizado esta primera parte para ello. Finalmente, debido a que su realización era análoga al número de alumnos, aceptamos y realizamos las modificaciones necesarias de esta primera parte.

4.5. Aplicación web

Como es lógico, a lo largo del desarrollo de la aplicación y de la implementación de los CMI nos han ido surgiendo multitud de problemas: errores de compilación, de ejecución, excepciones que no habíamos capturado, comportamientos indeseados, casos particulares que no habíamos tenido en cuenta y cuya ocurrencia es posible, etc. A lo largo de este apartado y del siguiente, a la vez que vamos comentando los distintos pasos que hemos ido siguiendo para la programación de la aplicación, haremos referencia a algunos de ellos.

Para empezar, creamos mediante NetBeans una aplicación web. Ya que esta debe de posibilitar al usuario una buena navegabilidad, pudiendo pasar siempre de cualquier página de la aplicación al resto, lo primero que realizamos es un fragmento HTML que incluiremos como cabecera en todas las páginas. Esta dispone de un enlace a la página de inicio de sesión en caso de que el usuario todavía no se haya autenticado y, en caso contrario, presenta la opción de cerrar la sesión. Como es lógico, para su realización usamos también CSS, así como también haremos para la creación de todas las vistas de la aplicación.

Ha habido bastantes ocasiones en las que no conseguíamos la presentación de la página que buscábamos. En estos casos nos ha resultado bastante útil ir modificando el CSS desde el propio Firefox ya que así íbamos viendo directamente qué cambios tenían lugar en la página, lo cual resultaba mucho más rápido.

A continuación creamos ya nuestra primera página de la aplicación, la cual se trata de la página de inicio. Esta presenta un aspecto u otro según si el usuario ha iniciado sesión o no. En caso de que todavía no se haya autenticado, esta tiene el aspecto presente en la figura 4.15. Y, en caso contrario, esta dispone de enlaces visuales a todos los cuadros de mando (ver figura 4.16).

Como hemos comentado anteriormente, todo usuario que desee acceder a los servicios de la aplicación web (cuya URL relativa es DatosMoodle) deberá previamente iniciar sesión. Por lo que si solicita ver cuadros de mando cuando todavía no lo ha hecho, se le redirigirá automáticamente a la página de autenticación (una vez haya iniciado sesión el principio de las rutas relativas de las páginas web serán DatosMoodle/Usuario). Para implementar el control de acceso realizamos lo siguiente. Creamos un jsp que contenga el formulario que queremos que se muestre al usuario para que este introduzca su usuario y su contraseña. Posteriormente creamos un servlet, cuya url asociada es DatosMoodle/Login, que realice un «forward» a dicho jsp en caso de que la petición que haya recibido sea de tipo get. Si en cambio esta es de tipo post, comprueba si el valor de los parámetros correspondientes al usuario y contraseña que se han introducido en el formulario son los indicados. En caso afirmativo, guarda dicho usuario en session y redirige al usuario a la página principal de la aplicación, de url DatosMoodle/Usuario. Y, en la situación opuesta, introduce en el scope request un mensaje de error y realiza un «forward» al jsp, el cual muestra la página de inicio de sesión pero mostrando el mensaje de error y el usuario escrito anteriormente.

Notar que no se han implementado medidas de seguridad en dicho control de acceso, como cifrado de contraseña, medidas para evitar inyección de código SQL, etc, ya que el cliente no le ha dado importancia a ello pues en un principio ni siquiera tenía pensado controlar el acceso a la aplicación. Aun así, considero que es una de las posibles mejoras que deberían realizarse en la aplicación de cara al futuro cuando realmente se vaya a poner en uso.

Figura 4.15: Página de inicio de la aplicación (/DatosMoodle).

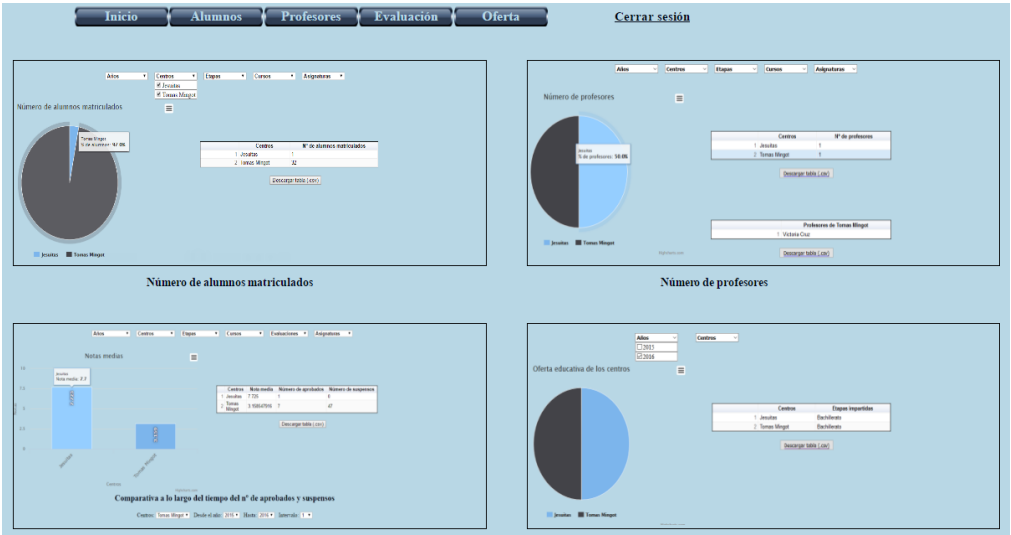


Figura 4.16: Página de inicio una vez el usuario se ha autenticado (/DatosMoodle/Usuario).

A lo largo de la implementación de toda la aplicación hemos ido teniendo en cuenta que el cliente desea una aplicación intuitiva y fácil de usar, por lo que hemos programado diversas medidas para conseguirlo. Un ejemplo de ello es la decisión de mostrar un mensaje de error cuando se autentique de forma errónea el usuario, conservar en ese caso el nombre de usuario que había introducido escrito en el formulario, poner el cursor automáticamente en la caja de texto en la que debe escribir, remarcar la opción seleccionada en el menú, etc. Debido a que estos son pequeños matices y mejoras de la aplicación, no nos pararemos a comentar todas en esta memoria, pero se podrán observar al utilizar la aplicación. Así como tampoco profundizaremos en todas las medidas de seguridad que hemos ido aplicando para evitar fallos o comportamientos inesperados: comprobar que los parámetros no son nulos previamente a su uso, longitudes de

arrays, capturar excepciones que se pueden producir, tratar todos los casos cuya ocurrencia es posible, como por ejemplo que el resultado a una consulta al cubo sea vacío, etc. Aunque sí cabe destacar que, para evitar repetir código, hemos creado dos filtros. Uno que comprueba siempre que el usuario pide una página que comienza por DatosMoodle/Usuario, que ya ha iniciado sesión y que si no es así, le redirige a la página de login. Y otro que se encarga de la correcta codificación de los parámetros.

Por otra parte, para el cierre de sesión, creamos un servlet que elimine el atributo usuario de la sesión, ya que mediante este distinguimos si el cliente se ha autenticado ya o no, y que le redirija a la página de inicio de la aplicación. Este servlet es llamado al hacer click en el enlace cerrar sesión de la cabecera.

Ahora pasamos a programar la conexión con el cubo, del cual obtendremos los datos a mostrar en los cuadros de mando. Tras formarnos, probar diversas maneras de realizarlo, obtener varios problemas referentes a rutas, versiones de Mondrian, uso de Connection o de OLAPConnection, etc, finalmente lo conseguimos del siguiente modo.

Creamos un fichero de nombre conexion.properties indicando el driverClass (el cual es en nuestro caso com.mysql.jdbc.Driver), la url referente al datawarehouse (jdbc:mysql://ip correspondiente:3306/datawarehouse), el usuario de MySQL y su contraseña, y la ruta donde tenemos el cubo a consultar. Una vez lo tenemos, creamos una clase con un método que contiene el siguiente código devolviendo así las propiedades anteriores.

```
try{
    Properties props = new Properties();
    props.load(getClass().getResourceAsStream("conexion.properties"));
    if (!props.isEmpty())       return props;
    else        return null;
}
catch (IOException ex){
        return null;
}
```

Posteriormente creamos una clase de nombre ConexionCubo que va a contener todas las consultas que necesitemos realizar al cubo. El constructor de ella, mediante el cual establecemos la conexión con el cubo, es el siguiente. Notar que esta clase consta de un atributo llamado prop de tipo Properties que contiene las propiedades anteriores.

```
try{
    String connectString = "Provider=mondrian;Jdbc="
    +prop.getProperty("url")+"?user="
    +prop.getProperty("user")
    +"&password="+prop.getProperty("pwd")
    +";Catalog="+prop.getProperty("rutaCubo")
    +";JdbcDrivers="+prop.getProperty("driverClass");
    con = DriverManager.getConnection(connectString, null);
}
catch(Exception e){
    e.printStackTrace();
    System.out.println("ConexionFallida");
}
```

En todos los servlets desde los cuales necesitamos utilizar algún método referente a consultas al cubo, declaramos un atributo estático de tipo ConexionCubo, al cual llamamos conex, y para evitar crear cada vez una instancia nueva de la clase ConexionCubo, le damos la siguiente implementación al método init().

```
if(this.getServletContext().getAttribute("conex")== null)
{
   this.getServletContext().setAttribute("conex", new ConexionCubo());
}
conex = (ConexionCubo) this.getServletContext().getAttribute("conex");
```

De esta forma, ya nos podemos centrar en la implementación de los diferentes cuadros de mando, para la cual hemos ido siguiendo lo comentado en el apartado 3.2.

4.6. Cuadros de mando

Debido a que el usuario debe poder seleccionar más de una opción en cada filtro y que además debe resultarle la aplicación amigable e intuitiva, tras barajar varias posibilidades e intentarlo de diversas maneras, nos decantamos finalmente por realizar para cada filtro, un select con checkboxes, tal y como se indicaba en [8] y que se puede observar a continuación. En los filtros referentes al cuadro de mando que muestra la comparativa a lo largo del tiempo del número de aprobados y suspensos por centro, al no poderse elegir más de una opción simultáneamente, hemos usado selects normales.

```
<div class="multiselect">
    <div class="selectBox" onclick="showCheckboxes(event)">
          <select id="centro">
               <option value="-1">Centros</option>
          </select>
          <div class="overSelect"></div>
    </div>
    <div class="checkboxes" id="opcCentro">
          <c:forEach var="c" items="${centros}">
             <div class="opc">
                <label for="${c}">
                   <input type="checkbox" name="centro" class="centro"
                    value="${c}" onchange="getInfoCentro()"/>${c}
                </label></div>
          </c:forEach>
    </div>
</div>
```

Estos checkboxes se despliegan al producirse el evento onclick sobre dicho select. En particular, la función JavaScript que se encarga de ello es la siguiente. Para realizarla, al igual que para implementar otras funciones JavaScript, y aclararnos con los parámetros que se pasan, con cómo acceder mediante HTML DOM a los elementos que queremos, etc, nos sirvió de mucha ayuda ir consultando mediante Firefox el valor de las variables y de sus propiedades.

```
function showCheckboxes(evento)
{
   var evt= evento || window.event;
   var src= evt.target|| evt.srcElement;
   var el = src.parentElement.nextElementSibling;
   if (!expanded) {
      el.style.display = "block";
      expanded = true;
   } else {
      el.style.display = "none";
      expanded = false;}}
```

Además, programamos también que cada vez que alguna de las opciones sea marcada o desmarcada (evento onchange), se vayan actualizando dinámicamente las opciones posibles a marcar en el resto de filtros de acuerdo a las que ya están marcadas. Esto se consigue mediante diferentes peticiones Ajax a los respectivos servlets, los cuales realizan las consultas pertinentes al cubo y actualizaciones del HTML correspondiente a partir del Json devuelto desde cada uno de los servlets con los datos. De esta forma evitamos que el usuario pida datos de una combinación inexistente, así como que solamente se puedan marcar cursos una vez está alguna etapa seleccionada. Esta actualización se va realizando en el siguiente orden, aunque solamente a partir del filtro al que corresponde la opción modificada: Años ->Centros ->Etapas ->Cursos ->Evaluaciones ->Asignaturas.

A continuación mostramos un ejemplo de petición Ajax que se lleva a cabo cada vez que se selecciona o se desmarca algún año. Como podemos ver, en este caso solamente es necesario pasarle al servlet GetCentros los años seleccionados. Además, para evitar problemas de timing, al no admitir la función success dos funciones distintas, decidimos crear otra auxiliar que invocase ordenadamente a las dos que debía llamar.

```
function getInfoAnio()
{
    getMarcados();
    $.ajax("../GetCentros",{
    data: {'anio' : anios},
    cache : false,
    responseType: "json",
    success : successAnio,
    onError : function (xhr, status, ex) {
        alert("Error("+xhr.status+"):"+msg);
    }
    });
}

function successAnio(data)
{
   actualizaSelectCentros(data);
   getEtapas();
}
```

Por otra parte, siguiendo con el mismo ejemplo, el método JavaScript que actualiza el HTML referente a las opciones posibles de centros a elegir es el siguiente. Inicialmente no habíamos tenido en cuenta que debíamos conservar las opciones marcadas previamente en cada uno de los filtros que se actualizaban, de manera que perdíamos esta información. Posteriormente lo arreglamos realizando un método que guardase todas las opciones seleccionadas e invocándolo antes de realizar la actualización. En el ejemplo siguiente la variable centros es la que contiene todos los seleccionados hasta el momento.

```
function actualizaSelectCentros(data)
{
    var content = "";
    var cts = centros.split(",");
    if (data.length != 0)
    {
        for (var i = 0; i<data.length; i++)
        {
           var marcada = false;
```

```
            for (var j = 0; j<cts.length; j++)
            {
                if(data[i] == cts[j])
                {
                    content += '<div class="opc"><label for="'+data[i]+'">
                    <input type="checkbox" value="'+data[i]+'" class="centro"
                    name="centro" onchange="getInfoCentro()" checked/>'
                    +data[i]+'</label></div>';
                    marcada = true;
                }
             }
             if(!marcada)
             {
                    content += '<div class="opc"><label for="'+data[i]+'">
                    <input type="checkbox" value="'+data[i]+'" class="centro"
                    name="centro" onchange="getInfoCentro()"/>'
                    +data[i]+'</label></div>';
             }
        }
    }
    document.getElementById("opcCentro").innerHTML = content;
}
```

Al haber varias similitudes entre los diferentes CMI a realizar, creamos un fichero JavaScript en el que incluimos para evitar repeticiones, métodos comunes a varios de ellos. Por ejemplo, los que actualizan a partir del Json recibido cada uno de los filtros, los que realizan las peticiones Ajax para actualizar los filtros, el que obtiene las opciones seleccionadas de todos los filtros, etc. Un fallo común ha sido olvidarnos posteriormente de añadir una referencia a dicho fichero u a otros o a librerías, en cada jsp donde los vayamos a utilizar. Así como referenciarlas erróneamente sin tener en cuenta la ruta donde nos encontrábamos, ya que copiábamos los enlaces de otras páginas en las que los habíamos utilizado, usar rutas absolutas en vez de relativas, etc.

Profundicemos ahora más en la parte de implementación relacionada con la presentación de la información en forma de gráfico o tabla. Tal y como comentamos en el apartado 3.2, usamos Highcharts para la creación de los gráficos (para lo cual nos han servido de mucha utilidad los ejemplos presentes en [4]) y Google Visualization para las tablas ([3]). Cabe destacar que esta última nos ha dado bastantes problemas: no encontraba dicha librería, posteriormente cuando descubrimos cómo referenciarla adecuadamente y que debíamos asociar su carga a la del document, indicando también la función JavaScript en la cual se utilizaría (habíamos intentado anteriormente hasta descargarla y usarla desde local), resultaba que había ocasiones en las que no le había dado tiempo a cargarse por completo previamente a su uso y por tanto solamente funcionaba haciendo Debugging y colocando puntos de interrupción. Para solucionar este último problema, nos decantamos por usar en alguna ocasión la función window.setTimeout o nos aseguramos que solamente era utilizada una vez se hubiese cargado por completo.

Con respecto a la creación de los gráficos, la hemos abordado de la siguiente forma. En todos los CMI hemos asociado a la carga del document la invocación de la función JavaScript getDatos(). Inicialmente, en cada CMI habíamos llamado al método que realizaba esta función con distintos nombres, lo cual nos daba problemas ya que algunos métodos comunes a varios cuadros encargados de actualizar los filtros, etc, debían invocarla. Así que finalmente, tras intentar también pasarle como parámetro a dichos métodos el nombre de la función a invocar, se nos ocurrió que esta era la mejor solución. A continuación podemos observar un ejemplo de la estructura de este método, el cual realiza una petición Ajax al servlet correspondiente, pasándole

a su vez las opciones seleccionadas para que este pueda saber y realizar las consultas necesarias y adecuadas al cubo, y posteriormente invoca a la función que crea el gráfico correspondiente en cada caso a partir del Json devuelto por el servlet. En particular, el ejemplo siguiente se refiere al CMI presente en la figura 4.17 que muestra el número de alumnos matriculados.

```
function getDatos()
{
    getMarcados();
    $.ajax("../GetNumAlumnos",{
    data: {'centro' : centros, 'etapa' : etapas, 'curso' : cursos,
    'asig' : asigs, 'anio' : anios},
    cache : false,
    responseType: "json",
    success : crearGrafico,
    onError : function (xhr, status, ex) {
        alert("Error("+xhr.status+"):"+msg);}
    });
}
```

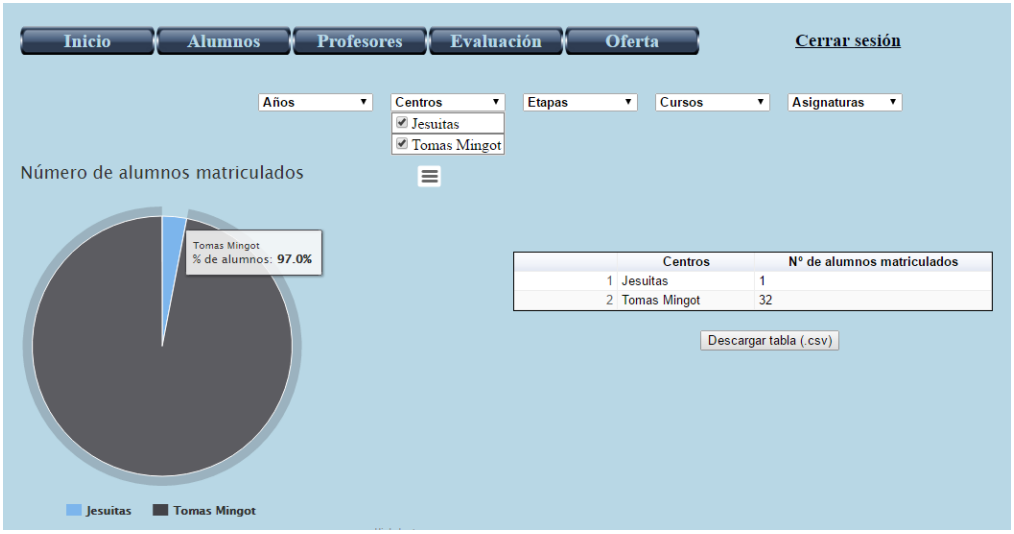


Figura 4.17: CMI referente al número de alumnos matriculados (DatosMoodle/Usuario/Alumnos).

Por otra parte, ya que en todos los CMI que contienen tanto gráfico como tabla, se deben actualizar y visualizar ambos simultáneamente, decidimos en ellos invocar a la función JavaScript que crea la tabla respectiva desde el propio método que crea el gráfico, pasándole como parámetro el Json devuelto desde el servlet. De forma que ese mismo servlet se encarga de devolver, en formato Json, una lista de Strings que contiene por un lado la cadena a utilizar para crear el gráfico y por otro, el String destinado a crear la tabla. Realizando esto nos encontramos con una gran cantidad de problemas que conllevaban que no se mostrasen los datos respectivos en los gráficos y tablas. Algunos de ellos fueron los siguientes. Problemas con las comillas: necesidad de escaparlas en ciertas situaciones y, a diferencia de en JavaScript, donde es indiferente colocar dobles que simples, en Json y Java solamente son válidas las dobles. Problemas con el formato Json: inicialmente pensábamos que dichos Strings se pasaban del servlet al jsp ya parseados a

Json, pero finalmente nos percatamos de que no, siendo necesario entonces aplicar a cada String la función JSON.parse(); por otra parte tampoco habíamos tenido en cuenta que dichos Strings debían seguir la sintaxis de Json, ya que solamente nos habíamos fijado en la sintaxis de los ejemplos que teníamos realizados con Highcharts y Google Visualization, de manera que al tratar de parsearlos a Json fallaba. Finalmente, gracias a un validador de JSON, fuimos averiguando la procedencia de los fallos y la sintaxis correcta que debían tener esos Strings.

A la hora de implementar los servlets, para evitar tener multitud de métodos largos y complejos muy parecidos, ya que para todos ellos vamos a tener que distinguir todos los casos posibles que se pueden dar, con respecto a opciones marcadas en los filtros, realizamos clases y métodos auxiliares. Estos, en determinados momentos, hacen una cosa u otra según cuál sea por ejemplo el valor pasado de un parámetro, el cual indica qué información se está pidiendo (número de alumnos, número de profesores, calificaciones finales...), realizan una acción u otra según si estamos en los CMI referentes al número de alumnos o de profesores, o de notas, etc. Cabe destacar que son tantos los casos posibles que se pueden dar, las diferentes opciones a realizar, etc, que para su correcta realización ha sido muy importante la fase de pruebas, ya que en ella nos hemos percatado de muchos fallos, casos olvidados o no previstos, etc.

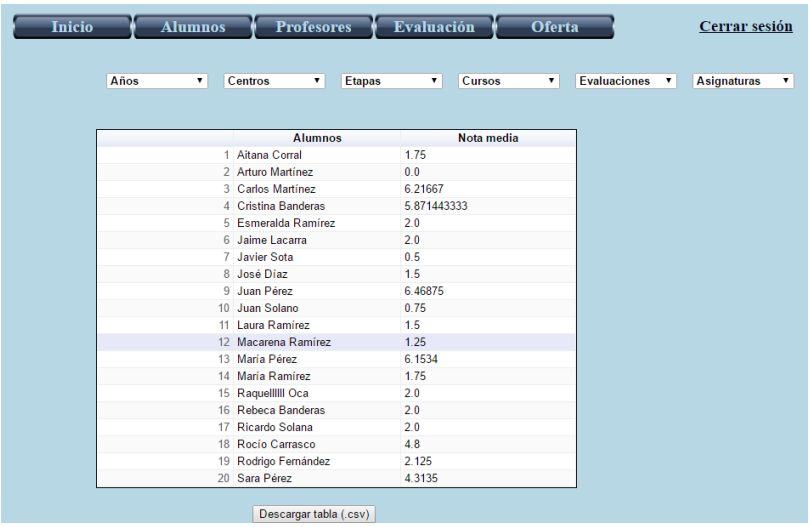
Con respecto a la implementación de que cada vez que el usuario hiciese click en una fila de la tabla del CMI referente a las notas o de la correspondiente al número de alumnos, le mostrase otra página con una tabla que contuviese los nombres de los alumnos junto a sus notas (ver figura 4.18), lo que tuvimos que realizar finalmente, tras buscar cómo se podía realizar y probar sin éxito de otras diversas maneras, fue añadir la siguiente instrucción «google.visualization.events.addListener(table, 'ready', mostrarDatos);» al método JavaScript que creaba la tabla, dándole a su vez la siguiente implementación a la función «mostrarDatos»: «google.visualization.events.addListener(table, 'select', mostrar);» y siendo finalmente la función JavaScript «mostrar» la que presentamos a continuación.

```
function mostrar()
{
  var selection = table.getSelection();
  for (var i = 0; i < selection.length; i++)
  {
      var item = selection[i];
      if (item.row != null)
      {
        var dimension = data.getFormattedValue(item.row, 0);
        document.getElementById("infoNotas").value = dimension;
        document.getElementById("cabecera").value = cabecera;
        document.getElementById("filtros").submit();
}}}
```

Al tener la nueva página que mostrar el menú en el que aparecían los filtros, con las opciones marcadas a las que la información que mostraba la misma correspondía, teníamos que pasarle también los valores marcados en el formulario del CMI del que procedíamos. Finalmente, tras barajar bastantes alternativas como por ejemplo hacer una petición GET al servlet de la nueva página a mostrar, añadiendo en el método «mostrar» la instrucción «window.location=Evaluacion/GetAlumnosNotas?» y después de la interrogación cada uno de los filtros con sus valores marcados, nos decantamos por lo siguiente. Como podemos observar en el anterior método, enviamos el formulario a dicho servlet (cuya url se indica en el action), habiéndole añadido anteriormente dos campos ocultos: uno para el valor de la fila marcada y otro para saber cuál el filtro principal al que refiere la información de ese CMI (asignaturas, centros...), ya que esta información la utilizamos para gestionar la lógica en el servlet. Notar que

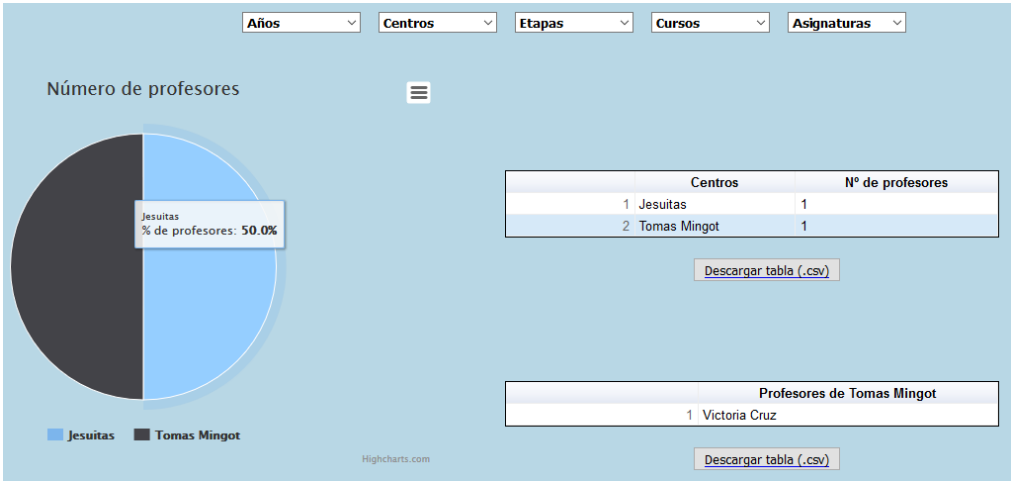
en esta nueva página los filtros son solamente de solo lectura, lo cual conseguimos previniendo el comportamiento por defecto de los checkboxes cuando se produce el evento onclick.

En cambio, en el CMI del nº de profesores (figura 4.19), al mostrar los profesores en la propia página, la función «mostrar» se encarga de realizar una petición Ajax, pasando como parámetros las opciones marcadas y el valor de la fila seleccionada, al servlet GetProfesores.



	Alumnos	Nota media
1	Aitana Corral	1.75
2	Arturo Martínez	0.0
3	Carlos Martínez	6.21667
4	Cristina Banderas	5.871443333
5	Esmeralda Ramírez	2.0
6	Jaime Lacarra	2.0
7	Javier Sota	0.5
8	José Díaz	1.5
9	Juan Pérez	6.46875
10	Juan Solano	0.75
11	Laura Ramírez	1.5
12	Macarena Ramírez	1.25
13	María Pérez	6.1534
14	María Ramírez	1.75
15	RaquelllIll Oca	2.0
16	Rebeca Banderas	2.0
17	Ricardo Solana	2.0
18	Rocío Carrasco	4.8
19	Rodrigo Fernández	2.125
20	Sara Pérez	4.3135

Figura 4.18: CMI referente a los alumnos y sus notas medias.



	Centros	Nº de profesores
1	Jesuitas	1
2	Tomas Mingot	1

	Profesores de Tomas Mingot
1	Victoria Cruz

Figura 4.19: CMI referente al número de profesores (DatosMoodle/Usuario/Profesores).

A continuación mostramos la apariencia que presenta el CMI referente a las notas, nº de aprobados y de suspensos, el cual a diferencia de lo planificado en la fase de diseño, contiene un gráfico de barras y no de sectores, ya que consideramos que así la visualización de las notas era más clara. Por otro lado, en la figura 4.21 mostramos aquel referente a la comparativa por años del nº de aprobados y suspensos en un centro. Para este último, debido a que nos basamos en la realización de los otros servlets que ya habíamos implementado, cometimos el error de que para

cada año, mostraba dos columnas que en vez de ser el número de aprobados y de suspensos de ese año, eran para un año el número de aprobados en ese año y en el siguiente, y para el otro (ya que solamente teníamos almacenados en Moodle dos años) el número de suspensos para cada uno de los dos años. Posteriormente en la fase de pruebas, nos percatamos y lo solucionamos.

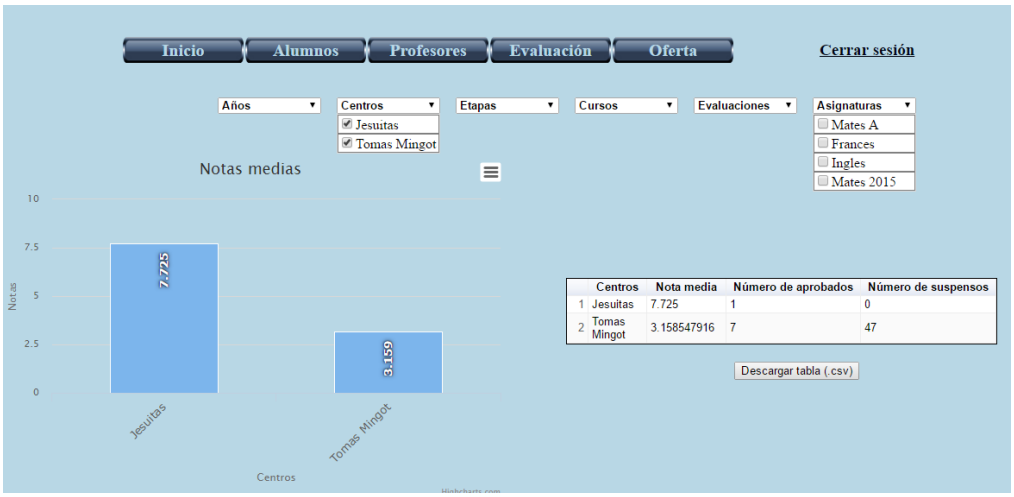


Figura 4.20: CMI referente a las notas, nº de aprobados y de suspensos (DatosMoodle/Usuario/Evaluacion).

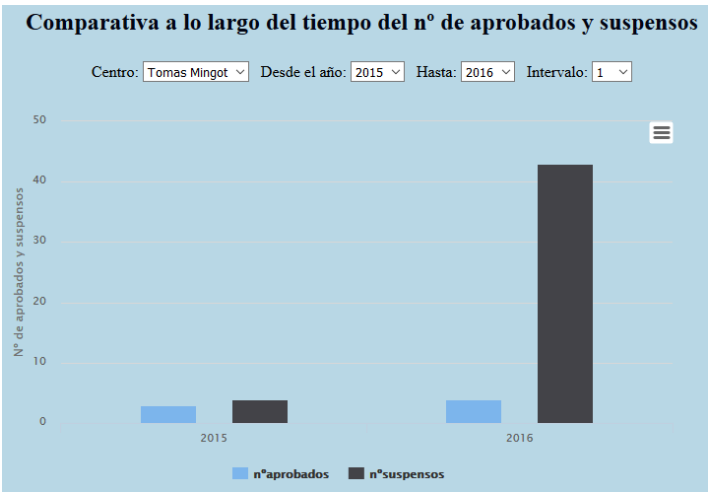


Figura 4.21: Comparativa a lo largo del tiempo del nº de aprobados y de suspensos en un centro.

Finalmente, debemos comentar que, en cuanto a conseguir que el usuario pueda descargarse todos los CMI, afortunadamente, al haber usado Highcharts para los gráficos, esta misma ofrece la posibilidad de descargarlos en distintos formatos. En cambio no nos ha resultado así de sencillo para las tablas, ya que a pesar de contar google.visualization con la función dataTableToCsv(data), hemos tenido que realizar ciertos cambios, como por ejemplo reemplazar comas por puntos y comas, pues no se mostraban correctamente los datos al abrirlos en Excel, formato que así requería el cliente.

Capítulo 5

Seguimiento y control

5.1. Alcance

Los objetivos alcanzados han sido los expuestos en el apartado de análisis de requisitos (apartado 2), salvo la parte opcional de despliegue de las notas de cada uno de los alumnos, a pesar de haberlo comenzado a implementar. Además, el requisito de que la aplicación fuese fácil de usar, intuitiva y pensando en lo más conveniente para el usuario, se ha satisfecho con éxito. Por otra parte, como extra, el usuario podrá descargar los gráficos no solamente en formato pdf, sino también en png, jpeg y svg, así como podrá utilizarla en diversos navegadores: Chrome, Firefox... a pesar de que al minimizar la pantalla se desajusten visualmente algunas partes.

5.2. Planificación temporal

Hemos llevado a cabo todas las tareas según aparecían en la estructura de la EDT. Sin embargo, se han producido ciertas desviaciones en algunas de ellas, con respecto al tiempo que les hemos dedicado o al periodo en el que las hemos realizado. En el cuadro 5.1 podemos observar, para cada tarea, el tiempo estimado, cuánto tiempo hemos dedicado realmente, y el porcentaje de desviación entre lo estimado y lo dedicado o realizado. Vamos a comentar ahora el motivo de las desviaciones más significantes.

La mayor desviación se ha producido en la tarea «Formación CMI» ya que no contábamos con conocimientos previos sobre ello y que en la realización de ejemplos, nos hemos tropezado con más problemas de los que creíamos que íbamos a tener. Algunos de ellos ya comentados a lo largo de los apartados 4.5 y 4.6: problemas configurando la conexión al cubo, referenciando librerías, dificultades para acceder a exactamente los valores que deseábamos de entre el resultado de una consulta al cubo, para crear tablas con Highcharts llegando finalmente a descartar dicha opción y a decidir realizarlo con Google Visualization, etc. Así como el estudio del CDE de Pentaho concluyendo que resultaba mejor y más sencillo programar todos los CMI y mostrarlos en la aplicación web, una vez aprendíamos a implementarlos.

La siguiente desviación de mayor tamaño ha ocurrido en la tarea «Creación cubo» debido a que ni en las prácticas ni anteriormente apenas habíamos visto cómo crear cubos, de manera que cuando realizamos la planificación no lo considerábamos tan sencillo, ya que realmente la dificultad reside en el proceso ETL. De hecho, en la tarea referente a este ha ocurrido también una desviación bastante significativa debido a esto y a los diferentes problemas e incidentes ocurridos durante el desarrollo de la misma, de los cuales hemos comentado ya algunos a lo

largo del apartado 4.2, y a que en general la base de datos de Moodle es bastante compleja.

	Tarea	Tiempo estimado	Tiempo real	Desviaciones
P1.1	Planificación	20 horas	22 horas	+ 10 %
P1.2	Análisis de requisitos	5 horas	5 horas	0 %
P1.3	Seguimiento	15 horas	11 horas	- 26.66 %
P1.4	Lecciones aprendidas	2 horas	2 horas	0 %
P1.5	Reuniones	15 horas	9 horas	- 40 %
P2.1.1	Estudio Moodle	5 horas	5 horas	0 %
P2.1.2	Introducción datos	5 horas	6 horas	+ 20 %
P2.2	Formación CMI	8 horas	20 horas	+ 150 %
P3.1	Diseño	10 horas	8:30 horas	- 15 %
P3.2.1	Construcción Data WareHouse	2 horas	3 horas	+ 50 %
P3.2.2	Proceso ETL	22 horas	32 horas	+ 45.45 %
P3.2.3	Creación cubo	20 horas	5 horas	- 75 %
P3.2.4.1	Aplicación web	20 horas	16 horas	- 20 %
P3.2.4.2	Implementación CMI	96 horas	102 horas	+ 6.25 %
P3.3	Pruebas	10 horas	14 horas	+ 40 %
P4.1	Manual	2 horas	1:30 horas	- 25 %
P4.2	Memoria	30 horas	34 horas	+ 13.33 %
P4.3	Presentación	13 horas	13 horas	0 %
	TOTAL	300 horas	309 horas	+ 3 %

Cuadro 5.1: Comparación tiempo estimado y real en cada tarea.

Además, se han producido también desviaciones bastantes significativas en la tarea P1.5 referente a las reuniones, ya que no ha sido necesario la realización de tantas como las que se planificaron puesto que la comunicación por correos ha sido excelente, en la tarea «Construcción Data WareHouse» por lo comentado en el apartado 4.1, y en la tarea «Pruebas» debido a lo trascendente que ha sido en este proyecto y a la gran cantidad de opciones, errores y casos posibles a tener en cuenta.

De todas formas, por lo general, las desviaciones producidas en las tareas no han sido muy grandes y además, afortunadamente, se han ido compensando mayores tiempos en unas con menores en otras, por lo que no han afectado mucho al desarrollo del proyecto.

Por otra parte, en las figuras 5.1 y 5.2, podemos ver los períodos estimados para cada tarea (gris) y los periodos de ejecución real de cada una (negro). Si nos fijamos, vemos que hemos ido siguiendo bastante la planificación, ya que era una forma de asegurarnos que íbamos a conseguir realizar el proyecto con éxito. Las mayores diferencias producidas son que por ejemplo, la tarea «Formación CMI» se ha ido realizando durante varias semanas aun posteriormente a la reunión de recogida de requisitos de la parte 2, cuando inicialmente estaba previsto haberla acabado para entonces; que la tarea «Lecciones aprendidas» la hemos realizado solamente al final de cada fase; que la tarea «Introducir datos» la hemos ido realizando a lo largo del proyecto para ir comprobando que el funcionamiento era el correcto, que no hemos ido realizando reuniones tan periódicamente como habíamos previsto, que el periodo de ejecución de la tarea «Proceso

ETL» ha sido considerablemente más largo de lo esperado pero que se ha compensado con el acortamiento del de «Creación cubo», etc.

Y a continuación, en el cuadro 5.2, se puede observar una comparación de las fechas estimadas y reales de los principales hitos del proyecto. Como se puede comprobar, la diferencia entre ellas es mínima, ya que durante el transcurso del trabajo nos hemos ido basando y guiando en las estimadas para asegurarnos que realmente íbamos bien de tiempo.

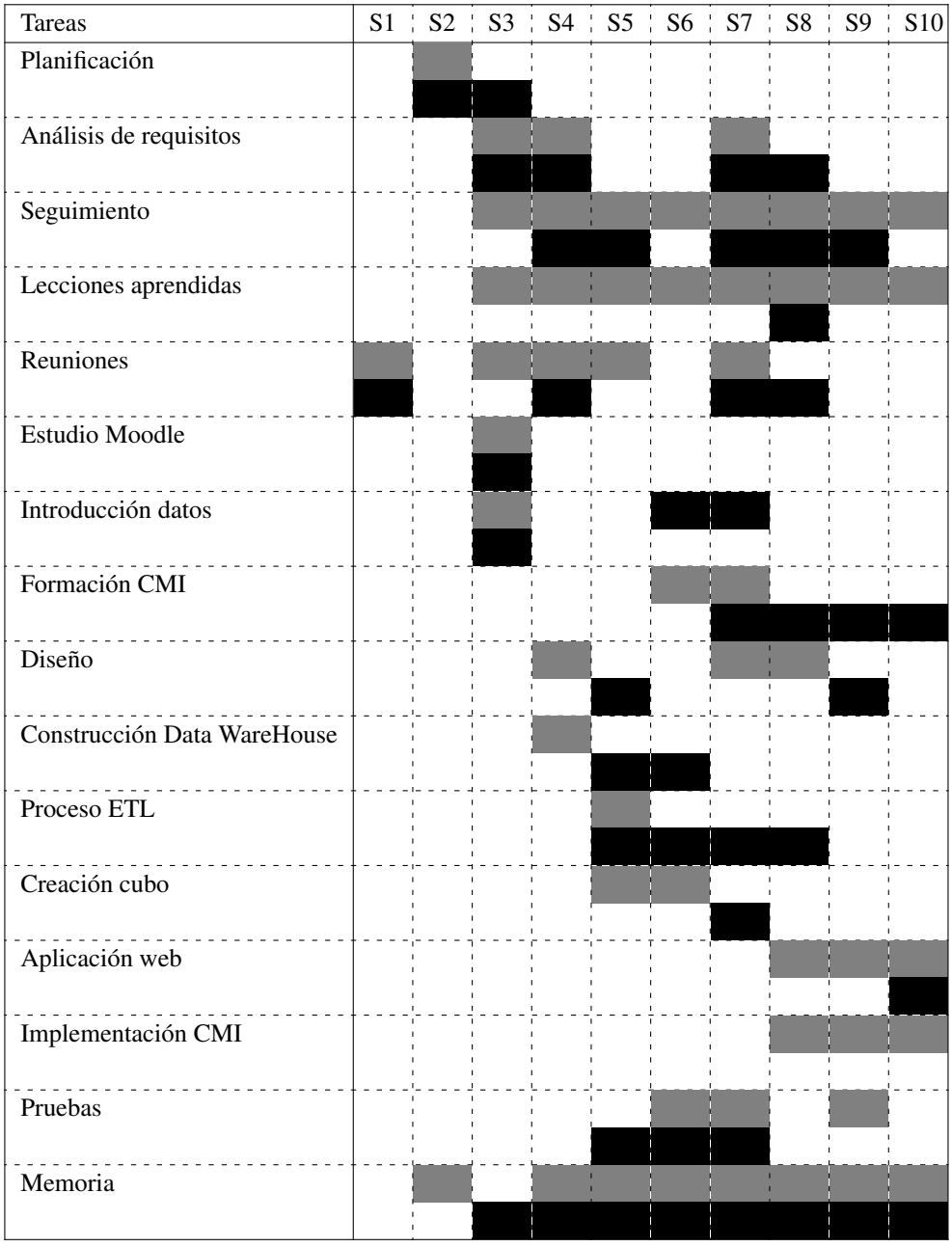


Figura 5.1: (a) Diagrama de Gantt (Semanas 1-10, siendo S1 la semana del 31 al 6 de Noviembre).

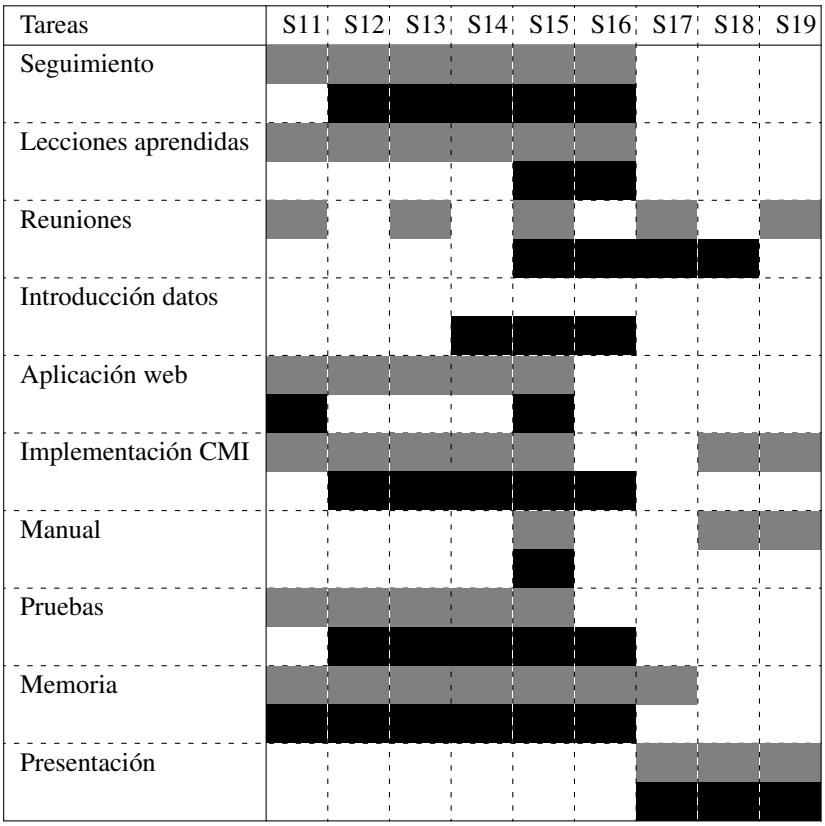


Figura 5.2: (b) Diagrama de Gantt (Semanas 11-19).

Hitos	Fecha estimada	Fecha real
Reunión inicio proyecto	Semana 1	Semana 1
Reunión aceptación requisitos parte 1	Semana 4	Semana 4
Reunión recogida requisitos parte 2	Semana 7	Semana 7
Reunión aceptación producto	Semana 15	Semana 16
Depósito proyecto	Semana 17	Semana 16
Defensa proyecto	Semana 19	Semana 19

Cuadro 5.2: Comparación fecha estimada y real de los hitos marcados (Semana 1: Del 31 de Octubre al 6 de Noviembre).

5.3. Seguimiento de comunicaciones

La comunicación con el tutor de este trabajo ha sido extraordinaria, tanto presencialmente como mediante correos en los cuales le enviaba también dudas o versiones de la memoria. Debido a ello, principalmente a que este me contestaba rápidamente al correo y me reenviaba la memoria escaneada con comentarios muy concisos y claros que él me había ido realizando, el número de

reuniones ha sido bastante menor que el esperado, como se puede observar en el Anexo B.

5.4. Control de riesgos

Los riesgos que han surgido y las decisiones tomadas sobre ellos han sido los siguientes. Afortunadamente, la ocurrencia de todos ellos los habíamos planificado ya (ver apartado 1.4), por lo que no ha alterado fuertemente el transcurso del trabajo.

- Cambios en las especificaciones o alcance por parte del cliente. En estos casos, hemos valorado los cambios y el impacto que conllevaban, y hemos aceptado solamente los que nos compensaban y resultaban viables para el desarrollo del proyecto. Además, si los cambios no se trataban solamente de pequeños detalles, los hemos aceptado negociando con el cliente recortes por otro lado en el alcance, de manera que se compensasen. Un ejemplo ha sido implementar que se descarguen todos los CMI en vez de realizar más CMI distintos, por lo que las horas destinadas a esta nueva tarea se han contado dentro de la tarea «Implementar CMI».

- La tecnología presenta limitaciones. Inicialmente habíamos previsto usar Highcharts para realizar tanto los gráficos como las tablas. Finalmente, debido a que analizamos esto anteriormente en la fase de «Formación CMI», decidimos usar para las tablas Google Visualization.

- Técnico. Nuestro ordenador no funciona correctamente, lo que provoca lentitud y colapso. De manera que tratamos de evitar tener abiertos simultáneamente varios programas, principalmente cuando se está ejecutando Pentaho. Además, nos surge también el problema de la memoria de la máquina virtual ya comentado anteriormente. Por otra parte, la esquina de nuestro ordenador se rompe y se abre completamente, de manera que evitamos transportarlo lo más que podamos para evitar daños mayores. Para prevenir y minimizar consecuencias, vamos realizando diferentes instantáneas en la máquina virtual, así como guardando copias de seguridad de todo lo referente a nuestro proyecto en dos discos duros externos.

Por otro lado, las potencialidades surgidas han sido las siguientes. También planificadas todas ellas en el cuadro 1.3.

- Buena sinergia con el cliente (planificada previamente), de manera que no nos ponía muchos impedimentos a nuestras propuestas, ni nos imponía de repente cambios bruscos en el alcance, etc. Por lo tanto, lo que hemos hecho ha sido tratar de mantenerla, dirigirnos a él de forma cercana aconsejándole y con educación, etc.

- Motivación por nuestra parte con respecto al proyecto (planificada previamente). Por lo que hemos aprovechado a trabajar en nuestros momentos más lúcidos y de mayor motivación al ver que íbamos avanzando tanto y que nos gustaba el resultado.

5.5. Control de calidad

Hemos ido marcando los requisitos presentes en el apartado 2 a medida que los íbamos satisfaciendo, de manera que finalmente, tal y como hemos dicho al comienzo de este apartado, el producto alcanza la calidad esperada.

5.6. Control de cambios

Con respecto a los cambios producidos, ha tenido lugar el cambio planificado en el cuadro 1.4 referente a la modificación de medidas, dimenensiones o filtros, ya que tuvimos que añadir la medida número de profesores, tal y como hemos comentado anteriormente. Este cambio ha tenido un impacto bajo en la realización del proyecto debido a su similitud con el caso número de alumnos matriculados, que ya habíamos realizado.

Por otra parte, también se ha producido el cambio previsto en la planificación correspondiente a añadir nuevas funcionalidades a la aplicación web, al pedirnos el cliente poder descargar todos los CMI. Pero a diferencia de lo que pensábamos, el impacto que ha conllevado ha sido medio y no alto, ya que supimos negociar con el cliente, en la reunión de recogida de requisitos de la parte 2, reduciendo el alcance acordado previamente (en particular dejando como optativa la parte de despliegue de notas comentada en apartados anteriores) a cambio de satisfacer este nuevo requisito. Además, en esta negociación también tuvimos en cuenta que realizar CMI una vez ya habíamos aprendido, nos resultaría más sencillo que dedicarse a la implementación de otra función distinta, como era el caso de descargar CMI, para la cual necesitaríamos formarnos previamente otra vez.

Capítulo 6

Lecciones aprendidas

Con la realización de este proyecto principalmente he adquirido bastantes conocimientos sobre BI y en particular, sobre la implementación de cuadros de mando integrales. Todo esto era totalmente nuevo para mi y desconocía la gran utilidad que presenta en la vida diaria.

A su vez, he aumentado mis conocimientos sobre diversos lenguajes de programación: JavaScript, JQuery, Json, Ajax, Java, HTML..., así como de LaTeX con la realización de esta memoria.

Por otra parte, debido a la gran cantidad de problemas a los que me he tenido que enfrentar, he acabado siendo capaz de realizarlo con gran velocidad y facilidad. Un ejemplo de ello es la capacidad para manejar con soltura y rápidez el Debugging, de cara a encontrar el lugar de procedencia de un fallo, tanto desde el propio Netbeans como desde Firefox en caso de que se trate de un error relacionado con JavaScript (botón derecho ->Inspeccionar elemento ->Depurador). En este último caso, ya que en NetBeans no se ofrece ningún tipo de información, se debe también consultar desde Firefox la consola en la que aparecen los fallos relacionados con el código JavaScript e incluso observar el código fuente que obtiene. A pesar de haber utilizado todas estas medidas previamente al proyecto, entonces no era tan consciente de la enorme eficacia que presentan.

Relacionado con la búsqueda y solución de fallos, cabe destacar también que me he percatado de que, a pesar de la enorme utilidad de Internet de cara a aprender cómo realizar cualquier aspecto relacionado con la informática o a conseguir solucionar fallos, hay muchísimas páginas en las que la información que se presta en ellas es errónea. De forma que no debemos fiarnos de lo primero que encontremos ya que esto puede traernos grandes pérdidas de tiempo, así como es muy importante conocer qué páginas son las más fiables o de mejor calidad. Por ejemplo, en este trabajo me ha sido verdaderamente útil la página Stackoverflow, ya que en ella se tratan multitud de aspectos informáticos, y de forma correcta.

Por otra parte, he aprendido que debo tener mucho cuidado con copiar y pegar código de otro jsp o servlet que hubiese realizado anteriormente, ya que en ocasiones hay que cambiar pequeños detalles. De manera que si uno no se da cuenta de ello en el momento, posteriormente pierde tiempo tratando de averiguar que el fallo procede de ahí. Por ejemplo me pasó esto al referenciar rutas, al copiar de un servlet a otro se me coló un simple espacio en la definición de un String, lo cual hacía que ya no se mostrase el gráfico Highcharts adecuadamente, etc.

Finalmente, tengo que decir que es trascendental pararse a pensar los métodos que vamos a necesitar implementar y su contenido, previamente a lanzarse a realizarlos. En alguna ocasión a lo largo del proyecto no apliqué esta medida y posteriormente lo sufrí, pues tuve que cambiar bastante de ellos porque no había tenido en cuenta ciertos aspectos.

Conclusiones

A lo largo de mi periodo de prácticas en la empresa Hiberus Osaba, mi tutor de la empresa me fue proponiendo varios temas para mi Trabajo Fin de Grado. En esa empresa habían realizado previamente para diversos clientes diferentes cuadros de mando, por lo que me ofrecieron esta alternativa. Como hasta entonces nunca había visto nada relacionado con BI, me mostraron algunos ejemplos y me explicaron en qué consistía. Finalmente, me acabé decantando por este tema ya que lo vi de mucha utilidad y consideré que era una muy buena oportunidad para aumentar mis conocimientos informáticos.

Posteriormente, discutimos sobre qué sistema de información sería mi fuente de origen de datos. Tras barajar distintas posibilidades, me decanté por los datos que se almacenan en Moodle, ya que el año anterior había cursado el Máster de profesorado y siempre me ha atraído e interesado todo lo relacionado con la docencia. Por lo que consideré que realizarlo a partir de dichos datos, además de motivarme más, me podría resultar útil en el futuro e incluso me parecía interesante disponer de la posibilidad de continuar con su estudio posteriormente a este Trabajo Fin de Grado. De hecho, este proyecto es bastante ampliable, es decir, tiene muchas líneas futuras. Para empezar, considero que una vez se vaya a poner en uso, sería muy interesante y conveniente implementar medidas de seguridad en el control de acceso y adaptar la aplicación a todo tipo de tamaños de pantalla, móviles, etc. Además, al ser la cantidad de datos distintos que se almacenan en Moodle tan grande, existen multitud de posibilidades de nuevos cuadros de mando interesantes a desarrollar, tanto a partir de las medidas y dimensiones ya presentes en el cubo, como a partir de otras tras realizar previamente el proceso ETL correspondiente, etc. Algunos ejemplos de estos son: despliegue de las notas de cada uno de los alumnos, es decir, la parte optativa cuya implementación ya está un poco comenzada y que hemos comentado en el apartado 2.2, diagrama de barras indicando el número de alumnos por rango de notas (para los filtros elegidos), gráficos que comparen diferentes centros en una misma etapa, ya que los que hemos realizado comparan los centros elegidos en su totalidad, etc.

Finalmente, haciendo una reflexión de estos meses de trabajo, la elaboración de este proyecto me ha resultado muy gratificante ya que me ha proporcionado muchísimas cosas. Además de obviamente haber aumentado en gran medida mis conocimientos sobre informática, pues además he tenido que utilizar bastantes herramientas y tecnologías distintas, he mejorado mis destrezas programando, desenvolviéndome, enfrentándome a errores o problemas..., he desarrollado la capacidad de aprendizaje autónomo, lo cual es tan importante en este mundo de la tecnología, he conocido realmente cómo funciona este mundo, cómo relacionarme con el cliente, etc.

En definitiva, a pesar de haber tenido que invertir tanto esfuerzo y horas de dedicación en este proyecto, me siento satisfecha de haberlo realizado y logrado ya que he aprendido muchísimo con ello y me ha hecho sentirme verdaderamente una ingeniera informática. Por lo que considero que tanto el desarrollo de las prácticas como del proyecto es imprescindible en el transcurso del Grado de Ingeniería Informática.

Anexos

Anexo A. Sistemas tradicionales vS Data WareHouses

Tal y como se dice en [2], las diferencias entre los sistemas tradicionales y los Data WareHouses son las siguientes.

Sistema tradicional	Data WareHouse
Predomina la actualización	Predomina la consulta
La actividad más importante es de tipo operativo (día a día)	La actividad más importante es el análisis y la decisión estratégica
Predomina el proceso puntual	Predomina el proceso masivo
Mayor importancia a la estabilidad	Mayor importancia al dinamismo
Datos en general desagregados	Datos en distintos niveles de detalle y agregación
Importancia del dato actual	Importancia del dato histórico
Importancia del tiempo de respuesta de la transacción instantánea	Importancia de la respuesta masiva
Estructura relacional	Visión multidimensional
Usuarios de perfiles medios o bajos	Usuarios de perfiles altos
Explotación de la información relacionada con la operativa de cada aplicación	Explotación de toda la información interna y externa relacionada con el negocio

Cuadro 6.1: Comparación de los sistemas tradicionales con los Data WareHouses.

Anexo B. Reuniones

A continuación presentaremos una breve descripción de cada una de las reuniones que hemos ido manteniendo a lo largo del desarrollo del proyecto tanto con el cliente, como con el tutor que nos ha ido asesorando y controlando la realización del mismo.

Reuniones con el cliente

Todas ellas tendrán lugar en la sede de Hiberus-Osaba en Logroño. De no ser así, se mostrará cuando hablemos de la misma.

Inicio proyecto

Fecha: 1 de Noviembre de 2016

Duración: 40 minutos.

Descripción: Primera toma de contacto con el cliente sobre el proyecto. El cliente nos cuenta lo que desea, la idea general del proyecto, sus dos principales objetivos (consultas de determinados datos de Moodle en función de distintos criterios y visualización gráfica de los resultados de las mismas), requisitos del proyecto, etc. De forma que a partir de esta reunión, podemos elaborar una planificación del proyecto (apartado 1), aunque no definir claramente los requisitos del mismo ya que nos pide investigar Moodle y a raíz de los datos que se almacenan en él, definir una lista de aspectos interesantes a estudiar y en función de qué variables, que le deberemos mostrar en la próxima reunión. Cabe destacar que nos comenta que solamente desea dejar fijados en el siguiente encuentro los requisitos de la primera parte del proyecto, es decir, los referentes a la parte de consultas.

Aceptación requisitos parte 1

Fecha: 25 de Noviembre de 2016

Duración: 30 minutos

Descripción: Presentación al cliente de la lista de requisitos recogida, referentes a la primera parte del proyecto. Modificación de la misma y aceptación final de la lista definitiva, la cual viene reflejada en el apartado 2.1.

Recogida requisitos parte 2

Fecha: 16 de Diciembre de 2016

Duración: 1 hora 30 minutos

Descripción: Muestra al cliente del producto elaborado hasta ahora en la primera fase del proyecto y aprobación del mismo. Discusión de los posibles requisitos interesantes de la segunda fase y finalmente, fijación de los mismos (ver apartado 2.2).

Aceptación producto

Fecha: 15 de Febrero de 2017

Duración: 1 hora

Descripción: Presentación, explicación y entrega del producto al cliente, y aceptación por su parte.

Reuniones con el tutor: César Domínguez

Todas ellas se darán en el edificio Científico-Tecnológico (CCT) de la Universidad de La Rioja, en particular en el Despacho 3234. En caso de que ocurra alguna excepción, se indicará al explicar la misma.

Inicio

Fecha: 3 de Noviembre de 2016.

Duración: 30 minutos.

Descripción: Explicación al tutor, de lo que nos ha comentado el cliente en la primera reunión sobre el proyecto. A raíz de esto, el tutor nos aconseja sobre la estructura de la memoria, finalmente decantándonos por una planificación para el proyecto completo y una división de las fases de análisis, diseño, implementación y pruebas en dos partes; una por cada objetivo principal del proyecto.

Fin primera parte del proyecto

Fecha: 22 de Diciembre de 2016.

Duración: 30 minutos.

Descripción: Informar al tutor de cómo está discurriendo el proyecto y del estado en el que me encuentro, así como presentarle qué haré a continuación. Comentar aspectos de la memoria y dar por finalizada por completo la primera fase del proyecto.

Prácticamente fin segunda parte del proyecto

Fecha: 9 de Febrero de 2017.

Duración: 1 hora

Descripción: Enseñar al tutor cómo va el proyecto y cómo lo voy sobrellevando, y consultar dudas y aspectos de la memoria.

Fin proyecto

Fecha: 15 de Febrero de 2017.

Duración: 1 hora 15 minutos

Descripción: Presentación final del proyecto al tutor, últimos aspectos a modificar de la memoria y autorización de depósito del proyecto.

Anexo C. Manual de configuración

Para el uso del producto, es conveniente que tenga en cuenta algunos detalles de configuración:

- Mediante un programador de tareas, deberá programar que se ejecute automáticamente la tarea de nombre: tareaActualiz.kjb.
- Indicar la dirección ip desde la cual accederá a Moodle. Para ello, modificar el archivo config.php que se encuentra dentro de la carpeta Moodle. Indicarlo también en todos los proyectos de Kettle. En caso de que cambie de ip o de contraseña, deberá actualizarla en dichos lugares.

- Indicar la dirección ip donde se encuentra el Data WareHouse y el cubo, en todas las transformaciones realizadas con Kettle, en Pentaho, en Workbench y en el fichero conexion.properties de la aplicación web. Por lo que si cambia de ip o de contraseña de MySQL, tendrá que actualizarla en tales sitios. En caso de modificar la ubicación del cubo, deberá cambiar también dicha ruta en el archivo conexion.properties.
- Una vez esté todo bien configurado, para acceder a Moodle bastará con poner en cualquier navegador: ip/moodle. Mientras que para usar Pentaho: ip:8080/pentaho (previamente se ha debido iniciar Apache, MySQL y Pentaho).
- Será necesario disponer de conexión a Internet, ya sea por wifi o cable. En caso de variar de una opción a otra, deberá cambiar la configuración de red de la máquina virtual. Obviamente, esta a su vez deberá estar iniciada para poder hacer uso de los recursos o servicios que contiene, así como los propios.

A continuación comentamos las versiones del software que hemos utilizado, con el objetivo de concienciar al cliente de qué debe instalar si desea utilizar el producto resultante de este proyecto en otro equipo distinto a la propia máquina virtual. Además, para hacer el despliegue de la aplicación, bastará con que introduzca el fichero de extensión .war generado, en cualquier servidor de aplicaciones con conexión a la base de datos.

- Pentaho 6.1. (Community Edition).
- MySQL 5.7.13 (y configurar MySQL como sistema de gestión de bases de datos que utilizará Pentaho).
- Saiku Analytics Plugin en Pentaho.
- Moodle 3.1, e importación de la base de datos correspondiente.

Bibliografía

[1] Anderson, Melissa. Tutorial cambio ubicación MySQL. https://www.digitalocean.com/community/tutorials/how-to-move-a-mysql-data-directory-to-a-new-location-on-ubuntu-16-04

[2] Fernández, Carlos. Foro sobre Data WareHouse. http://www.dataprix.com/que-es-un-datawarehouse

[3] Google Corporation. Gráficos Google-Visualization. https://developers.google.com/chart/interactive/docs/gallery/table

[4] Highcharts. Gráficos. http://www.highcharts.com/

[5] Joss, Jack. Blog Sistemas Informática y Electrónica. https://jossjack.wordpress.com/2016/03/18/pentaho-community-introducion/

[6] Lasanta, Inés. Pagina sobre CMI. http://www.iedge.eu/ines-lasanta-que-es-en-que-consiste-el-cuadro-de-mando-integral-cmi

[7] Reinoso Rojas, Ignacio. *Explotación de un Data Warehouse. Fundamentos y caso práctico para la gestión de proyectos.* Proyecto fin de carrera, Ingeniería de Telecomunicación, Universidad de Sevilla, Septiembre 2014.

[8] Stackoverflow. Foro sobre aspectos informáticos. http://stackoverflow.com/questions/17714705/how-to-use-checkbox-inside-select-option

[9] Universidad Alejandro de Humboldt. Blog sobre Ingeniería del Software. https://ingenieriadelsoftwareuah2015.wordpress.com/2015/04/05/inteligencia-de-negocios/

[10] Vera García, Antonio Carlos. *Análisis de herramientas BI en el mercado actual.* Trabajo fin de grado, Universitat Oberta de Catalunya, Enero 2015.